Computational Thinking in Augmented Reality: An Investigation of Collaborative Debugging Practices

Cheng-Yu Chung
CIDSE
Arizona State University
Tempe, AZ, USA
Cheng.Yu.Chung@asu.edu

I-Han Hsiao
CIDSE
Arizona State University
Tempe, AZ, USA
Sharon.Hsiao@asu.edu

***Abstract*—The uniqueness of Augmented Reality (AR) is its affordance to support learning abstract concepts by rich information, visualization and the integration of user-content interaction. Research has shown that abstract idea and invisible phenomena can be learned better with the support of AR. However, high complexity and conceptual Computational Thinking (CT), such as algorithm design and related programming concepts, are rarely studied empirically in the intersection with immersive technology. This study is aimed at addressing this issue and providing a piece of quantitative and empirical evidence to AR-supported Computer Science discipline-based learning. We designed a mobile AR-enabled application based on a CT framework and related AR affordances in the literature. This app can contextualize a programming debugging task and support program editing & CT learning. A controlled laboratory study was designed and conducted. The result of statistical analyses shows that participants with the AR support made better quality of programs with lower errors and less amount of code edits, compared to those without the AR support.**

***Index Terms*—augmented reality, debugging practices, learn to code, computational thinking, computer science education, collaborative learning, mobile app**

I. Introduction

Educational augmented reality (AR) refers to AR visualization designed for educational purposes [1]. It has been researched and used in learning activities in fields of science, technology, engineering, and mathematics (STEM). Such an application usually emphasizes the value of AR technology regarding its ability to impose an extra (or hidden) layer of information on physical surroundings. From the view of psychologists, this feature can effectively decrease the cognitive load during learning and therefore improve the perception of learning content that involves complex, abstract, and unobservable phenomena [2]. For example, in electrical engineering, AR can be used to visualize how magnetic fields evolve in a real speaker and affect the formation of sound. Therefore, a student can avoid extraneous cognitive load caused by the process of matching their imagination and observations they would have experienced in a traditional setting (e.g., 2D pictures) [3].

Although it has been shown that AR technology has the potential in support of learning complex concepts in many STEM fields, less explored is the use of AR for learning programming in Computer Science. When we introduce a programming concept to novice students, one major challenge is the alignment of Computational Thinking (CT) concepts, e.g., abstraction, algorithm, iteration, etc., and expected program outcomes. It is common to see that even when a student has read a program thoroughly, he/she is still struggling to understand how the program produces a desired output. This process is similar to learning content of other STEM fields which requires students to understand how a visible phenomenon is driven by invisible facts (e.g., the strength and range of magnetic fields that power Maglev; combination of hydrogen and oxygen atoms that form water, etc.). We believe that AR has its unique ability to solve similar instructional challenges in learning and teaching CT.

Both essential components in CT and unique affordances of AR have been reported in vast research literature [2], [4]. However, to the best of our knowledge, we have not seen many empirical studies that incorporate frameworks from the two parts and design and evaluate how such a combination can help (or hinder) student learning in CT topics. Specifically, relevant questions, like how student behavior in designing an algorithm and group collaboration in a CT practice, can be supported by AR visualization have not been answered yet by quantitative evidence of controlled laboratory study. The research goal of this study is therefore to fill this gap and develop the connection of CT and AR. We aim to investigate how to leverage AR affordances in CT learning and evaluate whether such a use can bring positive or negative effects to a CT problem-solving task. To this end, our research questions (RQs) in this study are:

1) How does the present of AR visualization affect participant performance of a debugging task in computational thinking compared to the non-AR visualization?
2) Does AR visualization support algorithm design in a debugging task? If so, what is the difference in code-editing patterns with and without the AR support?
3) Compared to non-AR visualization, how does AR visualization affect communication within a dyad in collaboration on a debugging task?

The remaining article is organized as follows: Section II collects and reviews related work of AR in STEM and the concept of computational thinking; Section III describes in

978-1-7348995-0-4/20/$31.00 ©2020 Immersive Learning Research Network
6th International Conference of the Immersive Learning Research Network (iLRN 2020)
Online, June 21-25, 2020.

details about an AR-enabled app for programming problem solving, principles in the design of task and AR visualization, and a user study for evaluating the work; Section IV presents our analyses which compare the AR visualization to the Non-AR one regarding patterns of code editing, code quality and user performance; finally the whole study is summarized and concluded in Section V.

II. Related Work

A. Augmented Learning Experiences in STEM Education

Much research has explored the effectiveness of AR learning experiences (ARLEs) in various STEM fields [3]. A distinctive characteristic of AR environment is the ability to allow the user to perceive virtual information projected on physical surroundings. For educational purposes, this feature can make learning material easier to comprehend by contextualizing the knowledge in a proper design [2]. Empirical studies have shown that ARLEs made students performed better when learning abstract concepts like mathematical thinking [2], electrical electromagnetism [5], or unobservable or invisible geographical phenomena [6]. Moreover, ARLEs may also have positive impact on student's affective states, e.g., attitudes toward learning, engagement, satisfaction, and motivation [7]. These studies suggest the feasibility of using ARLEs in STEM curricula not only for improving academic performance but for designing new kinds of learning activities that promote STEM education.

Although AR technology may be beneficial to certain learning activities, there are still some challenges and unresolved criticism in ARLE research. First, there are numerous kinds of ARLE designs and accompanied evaluations in literature. Such a design can be roughly categorized by three aspects: the hardware, the way AR visualization is presented, and the instructional design. Regarding the hardware, researchers might use devices like ordinary mobile phones and tablets (e.g., [6]), special head mount displays (HMDs) which are supplied by only few manufacturers nowadays (e.g, [5]), or special equipment which is adapted to certain environment (e.g., [8]). Regarding the way AR is presented in software, there are marker-based AR, markerless AR, and location-based AR [7]. Regarding the instructional design, there are discovery and exploration [9], problem-based learning [6], or pure lab-based experiment [5] to name a few. In addition, measurements employed in evaluation of AR studies are also diverse, especially when it comes to the measure of affective outcomes [7]. For example, "engagement" and "positive attitude" assessed may have subtle difference in their definitions and how they are quantified in one study from one another. This nature of ARLE research may show the versatility of AR technology when applied to learning activities. However, it makes the comparison between different studies difficult due to heterogeneous research settings and methods [7].

Another common criticism for ARLE is the comparison with "3D visualization" which refers to conventional 3D computer graphics presented on 2D computer monitors. To some extent, we believe AR visualization can be seen as 3D visualization projected on physical surroundings. Unlike virtual reality (VR) which is popularized by its full immersion in visual effects of learning experience, AR seems to lack in a theoretical basis that supports how AR is superior to conventional 3D visualization. For example, in an extreme case, one may argue that placing a computer monitor next to the material to learn is actually an "AR experience", which is probably not incorrect in some sort. However, we believe what makes AR different from conventional 3D visualization and VR is not simply rendering virtual information on physical surroundings but the dynamics in the user-content and user-user interactions. The effect from this "physical dimension" [2] is less explored and examined in the existing literature.

It is hardly feasible to address the issue of heterogeneous research designs (which is also probably out of scope of single research paper). Nevertheless, we believe it is practical to ground our research on a design of ARLE that is more likely to be replicated and comparable to future studies. Therefore, this study is aimed at providing a piece of empirical evidence regarding the effects of *marker-based* AR visualization on student problem-solving procedure on an ordinary *mobile tablet*, assessed by *quantitative log data*. We specifically designed the ARLE to leverage not only user-content interaction, but also *co-located between-users interaction* (see Section III for detail), which we believe is what VR and conventional 3D visualization lack of.

B. Computational Thinking and Debugging Practice

Although there have been much research focusing on the use of AR technology in STEM, to the best of our knowledge, there is few ARLE research focused on computational thinking (CT). Deeply rooted in fundamentals of computer science education, nowadays CT has evolved from the original idea about developing solutions which can be operated by computer agents [10] to the incorporation of social aspects in computer-supported learning activities [11], [12]. There have been many instructional designs based on this concept, e.g., project-based learning [11], pair programming [13], physical computing [14], etc. In recent years, we have also seen more effort on applying the model of CT to domains other than Computer Science [4].

Researchers have proposed numerous CT frameworks that encompass epistemological facets of CT. Despite of subtle differences across frameworks, *decomposition*, *abstraction*, *algorithms*, *debugging*, *iteration*, and *generalization* are commonly found across CT frameworks [4]. All of these facets or a part of them are more or less involved in the aforementioned CT instructional designs. It is worth noting that among these facets, debugging is a comprehensive problem-solving activity which incorporates all the other CT facets because, to successfully complete a computer program, one must be able to decompose the problem and apply different algorithms on sub problems iteratively without syntactic and semantic errors. Such a problem-solving procedure also requires a certain practice in CT learning [11].

To the best of our knowledge, there is few research focused on the use of AR technology in CT learning. Similar to other STEM fields, CT learning also involves concepts that are abstract and observable. For example, the algorithm steering an autonomous car is not visible from the outside. The debugging practice is also abstract in the way that it requires the learner to observe patterns of program outputs, trace the control flow, and then model the execution of the correct program. We believe such an activity can potentially be benefited by AR technology considering that AR visualization can support the user align an observation and his/her mental model of it (such a procedure is called *spatial and temporal contiguity* in [2]). Therefore, this study is aimed at filling this gap by applying AR technology to debugging practices of CT and evaluating what are effects of AR on the problem-solving procedure.

III. Methodology

Although there have been many efforts in either visual programming languages or AR in STEM education, to the best of our knowledge, there are few implementations in the literature investigating the use of AR technology in the context of CT. To fill the gap regarding how this new technology can be used in CT, we followed our previous work [15], [16] and designed a mobile app that combines these two concepts together. This section describes in detail about the design of the AR application, highlights a unique feature that promotes group collaboration in problem-solving tasks, and a controlled lab study for evaluation.

A. Design of Debugging Tasks in Programming for Mobile Devices

A conventional programming task involves much coding work which requires a keyboard. The complexity of regular programming language also needs a large workspace on screen to display all necessary components for users to design and execute a computer program. However, the limited interface of mobile device, like the tablets on which we developed our AR-enabled app, rarely has the ability to meet these specs. Simply side-loading a conventional programming environment onto a mobile device is impractical and may result in extraneous burden on the user that may inevitably reduce the quality of problem solving.

To avoid limitations of the interface and develop a proof of concept in this study, we followed the design language of block-based programming tool and simplified the programming interface to fit the small workspace on a mobile device as shown in Fig. 1. A script of code is displayed in blocks of command on the right programming panel. The task instruction and content are shown in the middle. When the run button is pressed, the program will execute the code from the top to the bottom sequentially.

This programming interface keeps away from the use of conventional syntax that may overwhelm the user in such a small screen. However, a task in this interface can still retain the complexity that requires users to have sufficient ability of CT to design a correct solution. In the task ”debugging an autonomous helicopter”, users are asked to fix a broken computer program and make sure it can bring the helicopter fly over walls and meanwhile collect all the flags on its way. This task was designed carefully based on essential components in an established CT framework including *abstraction* (extracting relevant information), *decomposition* (breaking down the problem into smaller parts), *simulation* (imitating a real-world process), *pattern recognition* (observing regularities), and *algorithm design* (creating an ordered series of instructions) [4]. The difficulty of the task was also tested and tweaked in a pilot study to ensure it can be finished in reasonable time.

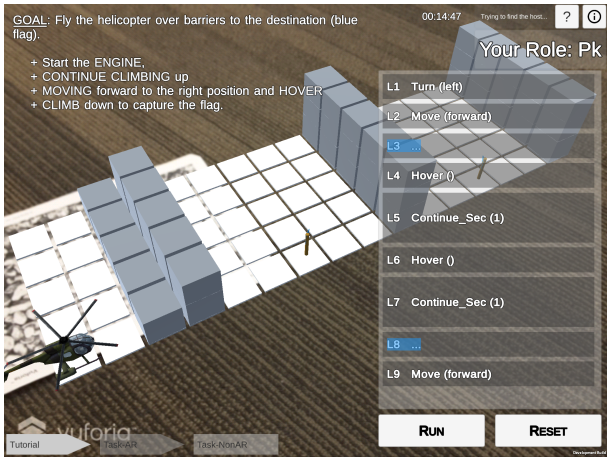

Fig. 1. A Screenshot of the App and Task in the AR Mode. The instruction is shown on the top which asks the user to capture the two flags in the map. The main visualization is positioned in the middle of the screen. On the right hand side it is the programming interface where the user can modify and test the code.

B. Displaying Program Outcomes in AR Visualization

The affordance of AR visualizations in educational contexts has been investigated and summarized in existing research. When designing the app, we followed a theoretical framework and made the AR visualization adhere to the debugging task by adapting the AR affordances *immersive visualization* and *spatial and temporal alignment* [3], [7]. The task in our app requires the user to observe and explore the geometry of virtual environment, and then connect it to algorithm design. To successfully complete the task, the user must associate the height and depth information with the movement of helicopter (e.g., Fig. 1). The user's mental image of object has to align with observations of outcome so as to design an algorithm that functions appropriately. This contiguity of spatial and temporal alignment is believed to help learners ”focus on the task at hand” [2].

Immersive visualization is realized in the interaction between users and content in the debugging task. In order to observe the detail and measure the depth and height, the user needs to approach the visualization and change the point of view. This action involves the user in exploration of digital assets and makes the activity more engaging than conventional media [3]. Although this format of visualization may never

Fig. 2. A Snapshot of Student Collaboration. The two students worked on their own devices but were allowed to freely communicate with each other in the task. The visualization was shared but the code snippets were different on the two devices. One student would not know what the other was doing/editing if he/she did not ask or share the screen. Since the execution of the two code snippets "took turns" (see Fig. 3 for details), the two students had to understand each other's code so as to make the program run correctly. This design is aimed at promoting collaboration among the users actively.

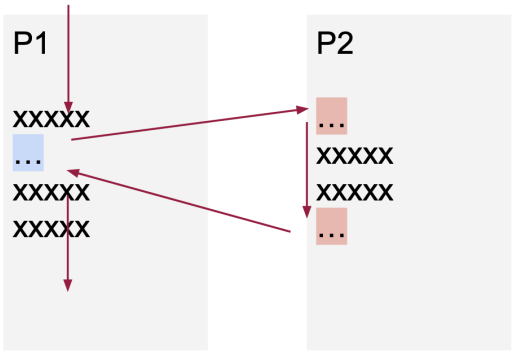

Fig. 3. An Illustration of the Code Synchronization in the App. The special "dots" commands are placed in code on the two devices (labeled by P1 and P2). Upon executed, the program is merged and works by taking turns. The arrows show how the control flows between the two code snippets.

be as immersive as a full virtual reality (VR) experience, we believe that this kind of immersive experience in AR is a good balance of user-content and user-user interactions which can immerse users in the task content to some degree but meanwhile do not deprive the interplay between users themselves in the workspace (e.g., Fig. 2). In addition, this constraint can probably be lifted by advanced AR head-mounted displays (HMDs). Nevertheless, discussion of hardware abilities and limitations is beyond the scope of this study.

C. Promoting Collaboration in a Dyad

Researchers have shown that collaborative learning "does not necessarily occur simply by putting two students together" [17]. They found that *turn-taking structures* in talk and *distributed productions* are important components in support of mutual understanding [17]. One research goal of this study is to investigate whether and how AR visualization affects collaboration between users in a debugging task. To this end, the design of our app and tasks has to promote collaboration actively. In other words, potential workload distributed among users should be balanced by the design itself.

We designed the task in the way that it cannot be finished by only one user in a dyad (see Fig. 2 for an example). The whole code is split into two parts on two devices. During the program execution, they are synchronized across the network. An illustration of this function is shown in Fig. 3. Since the total number of commands on one device is fixed and the synchronization command cannot be removed, this design reduces the likelihood of imbalance of workload in a dyad. The two users have to contribute more or less on their own and rely on the other's algorithm to achieve the goal. Note this design does not prevent the case that one user may directly take over the control of the other's device. Yet, this issue can probably be addressed accordingly by a instructional design and was also considered in the evaluation of this study.

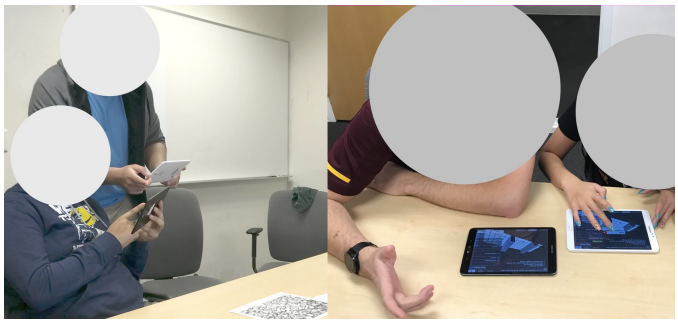

Fig. 4. Two Snapshots Showing How Participants Worked in the AR (Left) and NonAR (Right) Task. In the AR task, the participants could move around to see different angles of the visualization. On the other hand, the point of view was fixed at one angle in the NonAR task.

D. User Study Design

We designed a within-subject user study to evaluate the AR app for investigating how AR visualization affects student problem solving and collaboration in the debugging task. Participants were paired into dyad groups and assigned to two experiment conditions (or treatment groups), *AR* and *NonAR*. The AR condition presented the task content in marker-based AR visualization and the NonAR condition did in non-AR 3D visualization. In these two conditions, two participants operated on their own tablets with network connection and tried to solve the debugging task in the same environment collaboratively (Fig. 4).

The counterbalancing was used to avoid systematic variation in the within-subject design. Participants were randomly assigned into one of the two task orders: *AR-NonAR* or *NonAR-AR*. Those in the first order would experience the AR condition first, and the others did the NonAR one first. Participants spent about 25 minutes in one experiment, 15 minutes for the first condition and 10 minutes for the second one. This design was based on the result of our pilot study with three dyads of participants where we observed how participants used the app, their reaction and feedback. We believed that this setup would help us alleviate potential practice and boredom effects in the

within-subject design.

At the beginning for one study session, participants were asked to fill a questionnaire assessing their background knowledge in computer programming. It was followed by a 10-minute tutorial/training on the app interface and a 25-minute experiment procedure. Afterward, the participants were asked to finish two questionnaires assessing the user experience and the system usability. A semi-structured interview was followed to understand in detail their collaboration of using AR in solving programming problem with their partners. In total, a study session would take one dyad group around 60 minutes to finish.

E. Data Collection

The study has four major sources of data: 1) questionnaires assessing participant background, user experience, and system usability; 2) a semi-structured interview assessing open questions and gathering in-depth feedback; 3) task log data and 4) audio recordings of discourse occurring during the problem-solving procedure.

The questionnaires were printed on paper for the participants to work on in a study session. The interview was hosted by the researcher at the end of a study session. The task log and audio data were all captured on the tablets by the app automatically. The task log data is a hard copy of app log file containing all events happening in the app, e.g., code snapshots and execution errors. This is the major source where we evaluated participant performance in the programming task. The audio recordings contain two participants' communication during the task. They were mainly used to evaluate how their discourse evolved over time and whether it was affected in different experiment conditions. Collecting both task log data and audio recordings all in one device ensures the consistency of the multimodal data and reduces the need of extensive workload for record transcription and alignment. In this paper, the evaluation was mainly focused on investigating task log data and audio recordings because the other data sources were not designed for the research questions presented here.

We conducted the user study with students from an undergraduate computer informatics course in the semester Fall 2019. Participants were paired according to their self-reported experience in programming. Genders in a dyad were also balanced. In total, there were 12 dyads (13 males and 11 females; 24 students). 9 of the 24 students had less-than-one-year, 8 had one-to-two-years, and 7 had more-than-two-year experience in computer programming. Following the design of user study, each of 12 dyads received both AR and Non-AR versions of tasks in their own 25-minute study sessions. Finally, 24 tasks (12 AR + 12 NonAR) were conducted.

IV. Evaluation Results

A. Evaluating Performance in the Debugging Task

We analyzed the performance of debugging task by two factors: correctness of code and how the code were edited in the task. When it comes to correctness, one obvious question to ask is how many tasks are completed successfully by the participants. Note "successful completion" here means that the two participants can design an appropriate algorithm and successfully collect the two flags in the map. Out of 12 dyads, only 5 of them were able to pass one task in their study sessions. Additionally, we found that they all passed the AR version. The low passing rate ($\sim 20\%$) implies that the debugging task in the app was nontrivial and demanded some amount of effort to pass. What interesting but not out-of-expectation here is that none of NonAR tasks was completed successfully by any dyad. One possible reason of this outcome is that the depth/height information provided by AR visualization might help the users to measure how long/high their helicopters flied over the terrain. This information is indeed necessary to design an appropriate algorithm in this particular task.

In addition to the result of completion, we further explored the quality of participants' answers by comparing to the standard solution and tried to explain how much "error" was in their answers. We used the edit-distance measure, longest common subsequence (LCS), to represent the "error rate" of an answer compared to a reference code. This error rate should be 0.0 if the two are identical and 1.0 if they are distinct.

Just like conventional programming tasks, the debugging task in our app has more than one possible solution due to subtle differences in implementations. However, we assume that all solutions should share similar high-level algorithmic designs since the goal is to move the helicopter in a straight line in a space and the total number of commands is capped. To reflect this assumption, we calculated *global error rate*, which measures the quality of high-level algorithmic design, and *local error rate*, which measures the correctness of detailed implementation. Namely, the global error rate measures the dissimilarity between a whole program code and a simplified and abstract solution; the local error rate measures the dissimilarity between parts of code and their own solutions, each of which is used to capture one flag in the task. We believe the two-level measurement can reveal fine-grained detail about how participants edited code in different visualizations.

The distributions of global and local error rates are shown in Fig. 5 and Fig. 6. Fig. 5 is rendered by average error rates of all post-edit snapshots in the tasks. This measurement gives an overview of problem-solving performance. Fig. 6 is rendered by error rates of the final post-edit snapshots, which can be treated as the quality of final answer.

In Fig. 5, we can see that the average global error of the AR group ($M = 0.43, SD = 0.06$) seems to be higher than the NonAR group ($M = 0.40, SD = 0.04$). This observation was verified by a significant difference in the paired sample t-test ($t(11) = 2.77, p = 0.02$). Yet in Fig. 6, the final answers from the AR group ($M = 0.36, SD = 0.05$) have much lower global error than the NonAR one ($M = 0.43, SD = 0.10$), with a significant difference ($t(11) = -2.28, p = 0.04$). Such two differences were not found in either the average local error or the final local error even though the AR group showed a trend where the error rate was lower than the NonAR one. These outcomes signal an evidence that participants solved

6th International Conference of the Immersive Learning Research Network (iLRN 2020)
Online, June 21-25, 2020.

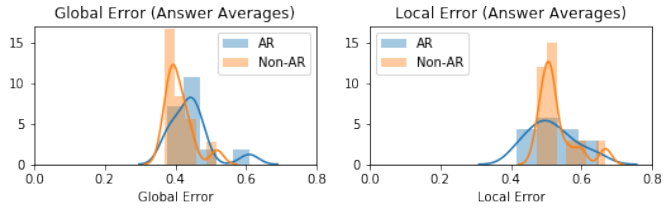

Fig. 5. The Distribution of Average Error Rate from the AR and NonAR Task. The values are averaged error rates of all answers (i.e., produced code after each edit) in the AR or NonAR task. On the left chart we can see the NonAR group has a little lower average global error than the AR one. But the average local error rates are quite similar in the two groups as shown on the right chart.

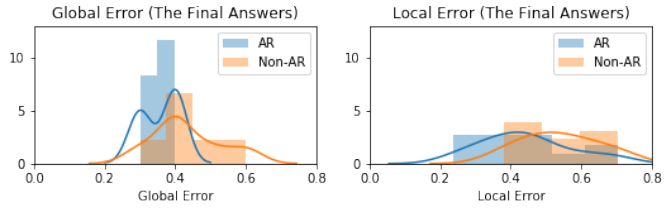

Fig. 6. The Distribution of Final-answer Error Rate from the AR and NonAR Task. The values are error rates of the final answers in the task. In both charts, we can see the trend that the AR group has lower error than the NonAR one.

the debugging task more effectively in AR visualization than doing in NonAR. Although in the AR task, the participants might have many versions of answers which resulted in higher average error, they were still able to produce better-quality final answers that achieved a much lower error rate than the NonAR group.

B. Extracting Ineffective and Redundant Editing Attempts

In addition to the correctness of code, we further inspected on a lower level about how the participants edited their code. There are two types of code edits in the debugging task: command editing (EDIT_CMD) and parameter tweaking (EDIT_PARAM). EDIT_CMD refers to the edit of commands that practically changes the behavior of helicopter in the task, e.g., changing from moving forward to climbing up. On the other hand, an EDIT_PARAM edit only tweaks the outcome of the same command. In the task, there is a specific command

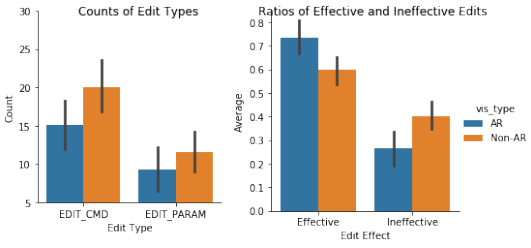

Fig. 7. Factoring Amount of Code Edits by Types and Effectiveness. The left plot compares the two types of code edits. The right plot compares the effectiveness of code edits, which only includes those increase or decrease the total error, i.e., the sum of global and local error rates.

"Continue_Sec" which accepts one integer as a parameter and continues the previous action for a given amount of time. To be specific, this command works like the loop operation in regular programming languages. The programmer's job here is to find the most adequate number of times this loop should run to solve the task. We counted these two types of command edits in the AR and NonAR groups and the result is shown in the left plot of Fig. 7.

We noticed that the total number of code edits in the AR group ($M = 25.25, SD = 6.15$) was less than that of NonAR ($M = 35.08, SD = 11.00$) in the statistical testing ($t(11) = -2.30, p = 0.04$). There was no difference found between AR and NonAR in both the number of EDIT_CMD and EDIT_PARAM. But we found that in the same group (i.e., within AR or NonAR), the number of EDIT_CMD seemed higher than EDIT_PARAM. Our statistical testing showed that only in NonAR, the number of EDIT_CMD was significantly higher than EDIT_PARAM ($t(11) = 4.15, p = 0.00$).

Superficially, these outcomes simply indicate that in the NonAR task, the participants tended to have more edits than the AR task. However, if we consider this result together with the analysis on the correctness of code that showed the AR group did better than the NonAR group, it is likely that the NonAR group might have too many redundant or ineffective edits than the AR group. Additionally, the within-group comparison showing the NonAR group had more EDIT_CMD than EDIT_PARAM suggests the possibility that participants in the NonAR task might not be able to concentrate their code design since they edited the EDIT_CMD commands more often.

To further investigate this possibility, we joined up the measurement of correctness and code edits by computing changes of error rates after edits and thereby categorizing code edits into *Effective Edit* (decreasing error), *Ineffective Edit* (increasing error), and *Neural Edit* (neither increasing nor decreasing error). When only considering the ratio of effective edits to ineffective ones (the right plot in Fig. 7), we found that the AR group has significantly more Effective Edits and meanwhile less Ineffective Edits than the NonAR group ($M = 0.26, SD = 0.13$ in Ineffective Edits of AR; $M = 0.40, SD = 0.11$ in Ineffective Edits of NonAR; $t(11) = -2.4, p = 0.03$). This outcome emphasizes that participants in the NonAR task did not design their algorithms effectively and efficiently. On the other hand, those in the AR task were able to use less amount of edits but keep more edits effective. As a supplement to this conclusion, the edit matrix showing all code edits from one command to another is attached in Fig. 8, where we can see the NonAR group has more diffusing pattern than the AR one.

C. Discourse Analysis on the Audio Recordings

One source of data in this study is the audio recordings of participants' conversation. This data provides another channel of information which serves as evidence of their real responses to treatments. To fully utilize this data, the researchers first transcribed all the audio files into text with the help of a speech-to-text application. Afterwards, they proofread the

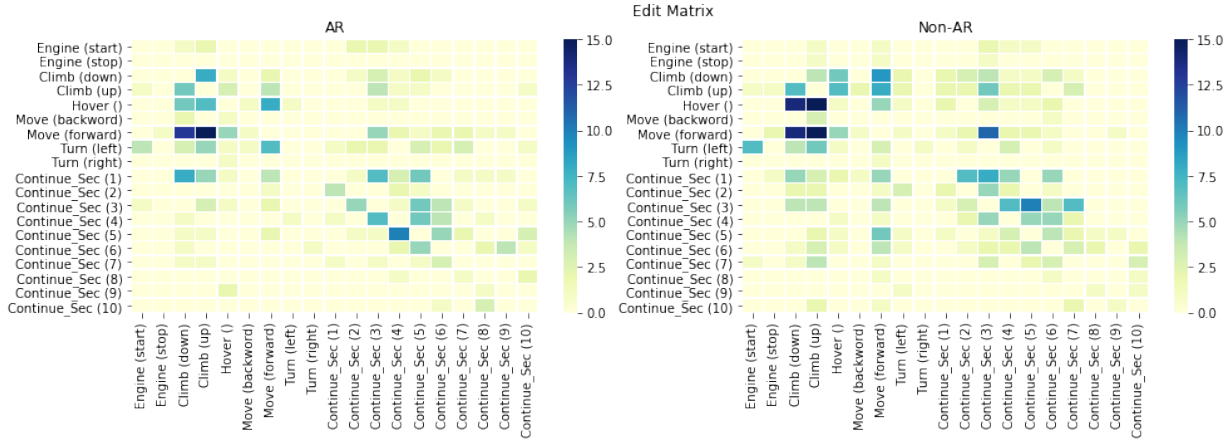

Fig. 8. Code Edit Matrix of All Possible Commands in the Debugging Task. Each cell stores the number of edits from the command on y-axis to the one on x-axis. This matrix provides an intuitive overview for comparing how participants edited their code in the task. In AR, participants did gradual and ordered edits to tweak the duration variable (see the bottom right on the left chart). But in NonAR, such edits seem to be messy and have multiple rounds. We can also see the edits in AR are more concentrated in certain places compared to the NonAR ones. This comparison suggests that AR may help participants to make more effective edits.

transcriptions and labeled all sentences by speaker tags to distinguish speakers. Then, the researchers aligned the time of transcriptions and system logs by an automatic computer script.

To look into whether AR makes any difference to participants' reaction, we measured the amount of discourse in the two groups by counting tokens without stop words in transcriptions, normalizing the amount by total task time, and then computing the rolling average. But this variable did not reveal any difference between AR and NonAR. That is to say, the total amounts of tokens in the AR and NonAR groups were similar. We guessed it could be due to some "events" through out the task that made participants communicate in different ways. So total amounts of tokens were averaged out to about similar values in the two groups.

Considering this possibility, we therefore grouped transcriptions into event windows that match some events in the debugging task. We chose to use "Debugging Time" referring to the duration when the users were fixing their code. It was identified by using the timestamps in the system log data which showed the moment when the user ran the program and until when the program raised an error. After partitioning the transcriptions into Debugging Time and the otherwise, we found a pattern between the AR and NonAR groups as shown in Fig. 9. In Debugging Time, the amounts of the two groups were very close. Yet in the non-debugging time, the AR group ($M = 1.47, SD = 0.32$) had significantly higher amount of tokens than the NonAR one ($M = 1.26, SD = 0.31$) ($t(11) = 2.77; p = 0.02$). In other words, participants in the AR task spoke much more than those in the NonAR ones. This result may indicate that AR visualization was more engaging than NonAR one. However, we did not find the same pattern in the amounts of keywords like commands for debugging tasks. To understanding discourse content, an appropriate language model is required. Relevant

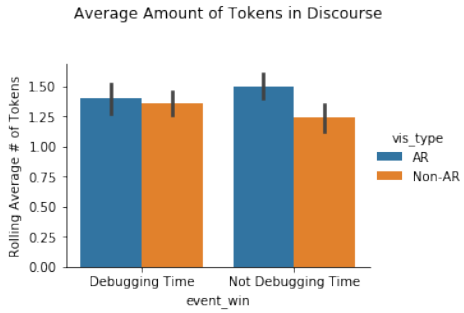

Fig. 9. Total Amount of Talk in the Two Event Windows. Debugging Time refers to the duration from an error occurs in the program to the next time the program is re-run.

research questions are left to be investigated in the future work.

To summarize, results from our evaluation indicate the trends that the AR group had higher passing rate of task, lower error rate in the code design, more effective code edits, and more communication than the NonAR group. These findings provide a piece of empirical evidence that AR technology has its potential in support of the debugging task of CT and brings a positive impact on algorithm design.

V. Discussion and Conclusion

Findings from our evaluation accordingly answer the proposed research questions RQ1 regarding the effect on task performance and RQ2 regarding the effect on algorithm designs. One caveat here is that these outcomes should be interpreted together with the design of AR content. We think that content presented in AR must be carefully designed in order to fully leverage the features of AR but not simply transferred from conventional media. Otherwise, the effects brought by AR may be mixed due to the mismatch between AR affordance and formats of content.

RQ3 asks how AR visualization affects communication within a dyad in collaboration. We believe that the way AR presents visualization may be a factor that can improve problem solving in collaboration since people can share the same workspace with extra layer of information to support them. In our discourse analysis, the audio recordings of participants revealed that the AR group talked more than the NonAR group in the non-debugging time. This result may indicate that AR was more engaging and the participants were willing to talk more. However, it is limited due to the lack of in-depth understanding of semantics in discourses within a dyad. We concluded that this RQ is only partially answered in this study and still requires further research to discover more evidence.

The result about task performance and communication in this work is aligned with other works in literature [7]. For example, in [5], Radu and colleagues found AR visualization was beneficial for students learning specific knowledge. In another example from [6], researchers integrated AR technology into the problem-based learning and found AR increased both learning outcomes and promoted positive attitudes. In addition to these general outcomes, this study contributes to the understanding of how AR affects the problem-solving procedure in a debugging practice of CT like the effect on algorithm design and group collaboration. Moreover, the measurement of evaluation in this work was based on real-time quantitative log data collected during the study session when participants were working on the task. Such a kind of data may reflect more closely the condition of user behavior during the task than a post-task assessment like questionnaire and interview. Finally, the AR application in this study was designed to use the marker-based AR and operate on an ordinary mobile tablet. Although lacking of advanced features like capturing the eye gazing and hand gestures and therefore the user interaction with virtual content is limited, such a design based on mobile tablet is more reproducible than the HMD-based design which requires specific AR-enabled hardware the is not accessible to the general public.

In conclusion, this study provides a piece of empirical and quantitative evidence contributing to the understanding of whether and how AR visualization affects problem-solving behavior in programming debugging tasks. We found that AR visualization can be helpful in supporting CT when the content is properly designed while taking AR affordance into account. A future study may replicate the design of AR application and the assessment applied in this study to make it more comparable across research settings.

ACKNOWLEDGMENTS

This research is partially supported by NSF CA-FW-HTF: 1936997.

REFERENCES

[1] I. Radu, "Augmented reality in education: A meta-review and cross-media analysis," *Personal and Ubiquitous Computing*, vol. 18, no. 6, pp. 1533–1543, 2014.

[2] K. R. Bujak, I. Radu, R. Catrambone, B. MacIntyre, R. Zheng, and G. Golubski, "A psychological perspective on augmented reality in the mathematics classroom," *Computers and Education*, vol. 68, pp. 536–544, 2013.

[3] M. E. C. Santos, A. Chen, T. Taketomi, G. Yamamoto, J. Miyazaki, and H. Kato, "Augmented reality learning experiences: Survey of prototype design and evaluation," *IEEE Transactions on Learning Technologies*, vol. 7, no. 1, pp. 38–56, 2014.

[4] V. J. Shute, C. Sun, and J. Asbell-Clarke, "Demystifying computational thinking," *Educational Research Review*, vol. 22, pp. 142–158, 2017. [Online]. Available: https://doi.org/10.1016/j.edurev.2017.09.003

[5] I. Radu and B. Schneider, "What Can We Learn from Augmented Reality (AR)?" in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. New York, New York, USA: ACM Press, 2019, pp. 1–12. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3290605.3300774

[6] M. Fidan and M. Tuncel, "Integrating augmented reality into problem based learning: The effects on learning achievement and attitude in physics education," *Computers & Education*, vol. 142, p. 103635, 12 2019. [Online]. Available: https://doi.org/10.1016/j.compedu.2019.103635 https://linkinghub.elsevier.com/retrieve/pii/S0360131519301885

[7] M. B. Ibáñez and C. Delgado-Kloos, "Augmented reality for STEM learning: A systematic review," *Computers and Education*, vol. 123, no. November 2017, pp. 109–123, 2018. [Online]. Available: https://doi.org/10.1016/j.compedu.2018.05.002

[8] J. Franz, M. Alnusayri, J. Malloch, and D. Reilly, "A Comparative Evaluation of Techniques for Sharing AR Experiences in Museums," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, pp. 1–20, 11 2019. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3371885.3359226

[9] T. H. Chiang, S. J. Yang, and G.-J. Hwang, "Students' online interactive patterns in augmented reality-based inquiry activities," *Computers & Education*, vol. 78, pp. 97–108, 9 2014. [Online]. Available: http://dx.doi.org/10.1016/j.compedu.2014.05.006 https://linkinghub.elsevier.com/retrieve/pii/S0360131514001286

[10] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, p. 33, 3 2006. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1118178.1118215

[11] K. Brennan and M. Resnick, *New frameworks for studying and assessing the development of computational thinking*. Vancouver, Canada: American Educational Research Association, 2012.

[12] Y. B. Kafai, "From computational thinking to computational participation in K–12 education," *Communications of the ACM*, vol. 59, no. 8, pp. 26–27, 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2975594.2955114

[13] H. Yuan and Y. Cao, "Hybrid Pair Programming - A Promising Alternative to Standard Pair Programming," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education - SIGCSE '19*. New York, New York, USA: ACM Press, 2019, pp. 1046–1052. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3287324.3287352

[14] G. Anton and U. Wilensky, "One Size Fits All: Designing for Socialization in Physical Computing," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education - SIGCSE '19*. New York, New York, USA: ACM Press, 2019, pp. 825–831. [Online]. Available: https://dl.acm.org/citation.cfm?doid=3287324.3287423 http://dl.acm.org/citation.cfm?doid=3287324.3287423

[15] C.-Y. Chung and I.-H. Hsiao, "An exploratory study of augmented embodiment for computational thinking," in *Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion*, ser. IUI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 37–38. [Online]. Available: https://doi.org/10.1145/3308557.3308676

[16] C.-Y. Chung, "Using augmented reality to support collaborative problem solving in computer debugging practice," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1426. [Online]. Available: https://doi.org/10.1145/3328778.3372693

[17] J. Roschelle and S. D. Teasley, "The Construction of Shared Knowledge in Collaborative Problem Solving," in *Computer Supported Collaborative Learning*, C. O'Malley, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 69–97.