Analyzing Reading Pattern of Simple C Source Code Consisting of Only Assignment and Arithmetic Operations Based on Data Dependency Relationship by Using Eye Movement

Shimpei Matsumoto[1(✉)], Ryo Hanafusa[2], Yusuke Hayashi[2], and Tsukasa Hirashima[2]

[1] Faculty of Applied Information Science, Hiroshima Institute of Technology, 2-1-1 Miyake, Saeki-ku, Hiroshima 731-5193, Japan
s.matsumoto.gk@cc.it-hiroshima.ac.jp
[2] Graduate School of Engineering, Hiroshima University, 1-4-1 Kagamiyama, Higashihiroshima, Hiroshima 739-8527, Japan

Abstract. Some programming learners in the lowest achievement group do not have even a minimum skill to read a simple program correctly. Reading programs would be an essential programming learning. To efficiently support learners in the lowest group, firstly we should conduct a fundamental analysis of reading programs to unveil their features. Therefore, the authors focused on eye tracking as a method to carry out the idea. The authors have thought that utilizing eye movement helps to clarify the reasons for making programming learning difficult. Therefore, the purpose of this study is to investigate the possibility of learner's program comprehension process based on the pattern of eye movement, not the eye distribution during reading source code. In this paper, we first measure the data of eye movement during reading some source codes and propose a modeling method to represent the feature of eye movement. Then we design an experimental protocol for analyzing eye movement based on program structure. The experiment of this paper focuses on source codes based on four types of data dependency relationship that can be generated by three lines of assignment statement only. As the analysis result, we confirmed that the data dependence of each pattern appeared as the unique eye behavior of program reading.

Keywords: Programming education · Reading · Eye tracking
Data dependency relationship

1 Introduction

Though programming has been regarded as a particularly important subject in the special field of higher education institutions such as universities, every year

S. Yamamoto and H. Mori (Eds.): HIMI 2018, LNCS 10904, pp. 545–561, 2018.
https://doi.org/10.1007/978-3-319-92043-6_44

there are many students who cannot accept any concept of programming. Sagisaka and Watanabe confirmed that most of the students belonging to the lowest group of programming comprehension level could not understand the most basic terms and grammars [1]. The existence of learners who cannot accept any concept of programming will be an especially serious problem when programming is integrated into the curriculum of compulsory education. Therefore, investigating when and why beginning programming learner gives up would be important; besides, programming education needs to establish an appropriate new instructional method for them.

Previous studies suggested that some programming learners in the lowest achievement group do not have even a minimum knowledge to properly read simple programs than writing a program consisting of dozens of lines. Reading programs must be an essential programming skill. Therefore, in programming introduction education, the authors think that supporting for reading programs must play an important role. Based on this idea, in order to effectively support learners in the lowest group, firstly educational experts including us should clarify their features from basic studies on analyzing program reading. Therefore, the authors focused on eye movement as a method to promote this idea. We thought that utilizing eye movement helps to unveil the reasons for making programming learning difficult.

This paper firstly examines whether the eye tracking is useful for estimating the reading process of programs by surveying and summarizing various results of previous studies. Based on the knowledge of these previous studies, the purpose of this paper is to investigate the possibility of learner's program reading comprehension based on the pattern of eye movement, not the eye distribution. This study fucuses on programs with only assignment and arithmetic operations, simplify the structure of programs to data dependencies, and set them as analysis targets. From the data of such programs obtained by eye tracking, this study constructs a simple Markov process model expressing eye movement patterns as transition probabilities. By comparing the difference of transition probabilities between nodes and data dependencies, this study shows the possibility of analyzing the program reading process based on eye movement.

2 Design of Analysis Method

2.1 Source Code Reading

The subject of this paper is to investigate and clarify whether the program structure is detectable by the pattern of eye movement under the assumption that programmer's thinking process appears on the eye movement. In addition to variable dependencies, actual source codes contain many elements not directly related to the internal structure, such as programming language specific writing style and algorithms. These are likely to be strongly influenced by experience. There is a suggestion that skilled programmers have abundant knowledge of problem domain and they use it efficiently [2]. As the first step, this paper aims to clarify that the program structure appears on the eye movement not depending

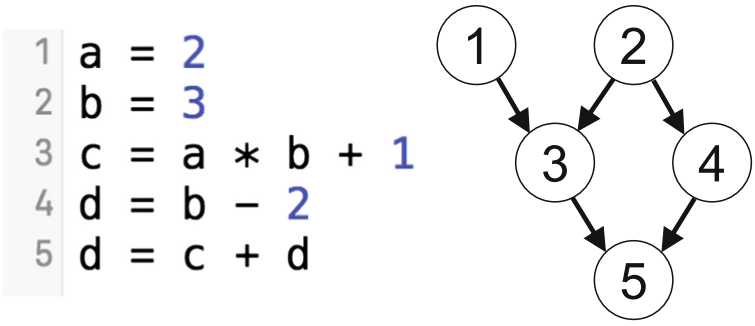
on the learner's experience. Therefore, actual source codes would be unsuitable for the experimental subject of this paper. Also, if an examiner creates the experimental tasks (source codes) manually, external factors other than the program structure may be included regardless of intention. Specifically, external factors are such as the design pattern of the program (variable name, the design of operator, and the description of assignment statements). These external factors should be eliminated as much as possible to clarify the eye pattern according to the program structure. Therefore, this paper does not generate the experimental task manually but automatically.

The source codes used as experimental tasks have no specific meaning, that is, has no purpose of processing. They are generated with arbitrary rules and depends only on the internal program structure. Also, they should be surely readable even if an examinee has just the most basic knowledge about the programming language specification and the minimum required calculation ability enough level as needed in daily life. Source codes not depending on the knowledge of problem domain will allow analysis without arbitrariness. Based on the suggestion of iterative learning of only sequential processing which is the basis of programming [3], even source codes including only a meaning of the internal program structure will be able to unveil the understanding process from the eye movement. In addition to these, to minimize the effect of memory ability, this paper designs an experimental protocol that uses source codes composed of concise instructions as the reading subject and allows examinees to read experimental subjects many times until examinees understand enough.

2.2 Flow of Analysis

The traditional analysis method of eye movement for program reading was based on time series data or the total amount of eye movement [4,5]. However, it is not easy to use the time series pattern as it is and to estimate the comprehension process of each examinee because the reading time and the reading strategy are entirely different depending on examinee. So this paper assumes that programmer's thinking behavior is similar to program slicing while reading a source code, and based on the assumption, grasps the behavior of program reading in terms of slicing. In other words, the assumption is that a programmer who understands the content of program correctly reads the source code while implicitly grasping the structure and the strength of data dependency relationship, and the comprehension process appears in the eye movement. Previous study qualitatively confirmed that the characteristic of the reading appears in eye movements depending on the program structure and the nature of programmer [6]. Therefore, this paper focuses on the fixation [7] which is a motion where gaze point stops within a range for a period and quantifies the time series fluctuation of the eye movement by fixation. Then, as a method for quantitatively analyzing the quantized data, a Markov model is employed.

The examinee's eye movement is measured during reading a source code until he/she grasps the values of all variables after the execution of the presented source code. Experiments in this paper do not limit the reading time



(a) A source code of C language (b) Program dependence graph

Fig. 1. An example of a source dode and its program dependence graph

and continue to measure until the examinee satisfied. As mentioned above, the experimental task is the source code of only simple assignment statements and includes only data dependencies among program structures (not including control dependency relationship). Thus, we believe that the history of eye movement can be evaluated not strongly affected by examinee's coding skill and reading strategy. Also, Markov model is useful for evaluating the pattern of eye movement because the examinee can repeatedly read and understand the presented source code many times until they fully understand the content.

2.3 Research of Analyzing Program Comprehension Process with Eye Tracking

Studies on eye movement targeting programming learning have also been conducted since the early 1990s [8]. Eye tracking instruments have been used to acquire nonverbal features of beginning/experienced programmers to analyze skill differences. However, although jumping of eye movement is common in source code reading, it is extremely rare in reading of natural language sentences [9]. Therefore, a unique method different from the analysis of general document reading is necessary to analyze programmer's cognitive process, but its analysis method is not sufficiently established even at the present stage.

3 Analysis Technique

3.1 Program Slicing

Program slicing is a technique that focuses on the dependency relationship between sentences in a program and extracts a set of sentences having dependency with a specific sentence, and the extraction result is called as a slice [10]. A slice is a part directly or indirectly related to information a programmer wants to know. The slicing technique has mainly used to narrow the sentence including

bugs in a procedural language and has achieved various developments such as static slicing and dynamic slicing [11]. Program slicing has also been reported to be important for debugging [12]. Therefore, an appropriate slicing skill will enable efficient work to quote existing sentences or to find bugs. Many learners have had a hard time finding a mistake in source codes. This work considered to be extraneous (inefficient) cognitive load. Proper slicing skill will enable programming learners to efficiently reduce non-essential work like scanning descriptive errors and to concentrate on essential learning like designing algorithm. As a means to quantify the slicing skill, the eye tracking would be effective. When reading a source code, we can assume that the learner implicitly makes a thought similar to slicing [13]. Although there are some studies focusing on slicing skill for programming education [14,15], the authors cannot find a research trying to estimate slicing skill from eye movement. If we can evaluate the slicing skill from eye movement, the analysis result will realize more appropriate instruction according to learner's skill level. Specific methods of program slicing include static slicing and dynamic slicing. The slice extraction process and its feature related to this paper are described below.

Static Program Slicing. In static program slicing, source code is analyzed statically and a dependency between statements is extracted. When the following conditions are satisfied for the statements s_1 and s_2 in the source code p, the statement s_2 depends on s_1, and the relationship is defined as the control dependency relationship.

- s_1 is the control statement.
- The result of s_1 determines whether s_2 is executed or not.

When the following conditions are satisfied, the statement s_2 depends on s_1, and the relation is called as data dependency relationship.

- In s_1 the variable v is defined.
- There is at least one route which does not redefine the variable v in s_1 to s_2.
- The variable v is referenced in s_2.

Program Dependence Graph. From the above dependency relationships, we can draw a program dependence graph (PDG) [16]. PDG is a directed graph in which edges represent dependency relationships between sentences, and nodes are statements such as control statements and assignment statements. The direction of the directed edge is expressed in the opposite direction to the direction of dependence in order to make it easy to grasp the flow of the program. An example of PDG is shown in Fig. 1. Figure 1(b) is a PDG created from slicing with the static slicing criterion (20 outputs) for the source code of Fig. 1(a). In Fig. 1(b), each node represents a row.

There are two types of dependency relationship, data dependency relationship, and control dependency relationship. In this paper, we focus only on the data dependency relationship. Programming skill to follow control syntax is considered to be highly dependent on knowledge. A learner who has not enough skill of control syntax would be difficult to follow the dependency relationship among variables involving the control syntax. Therefore, in this paper, we focus only on the data dependency relationship composed only of the most fundamental dependency relationship between variables. Figure 1 shows an example of data dependence graph in which a data dependence is expressed as a directed graph.

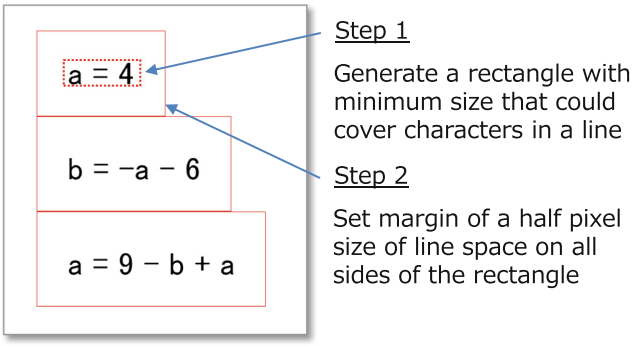


Fig. 2. Definition of the region of each line

3.2 Eye Tracking Technique

The movements of human's eyes when reading a document is roughly divided into two patterns [17]. One is a behavior called "Fixation" that gazes one point for about 100 to 500 ms, about 200 to 300 ms on average. The other is a quick movement called "Saccade" of about 30 ms that is a move from fixation to the next fixation. Saccade can be further categorized according to its characteristics such as the behavior of reading forward sentences (forward-reading) and the behavior of returning to previously read sentences (regression).

Measured raw data of eye movement contains many noises. To efficiently analyze the data of source code reading and evaluate these, only the characteristic patterns of eye movement while reading is considered to be available. Therefore, previous studies have usually observed only the history of the transition between each line a programmer gazed. Salvucci et al. proposed three methods to collate eye tracking data and process model: target tracing method, fixation tracing method, and point tracing method [18,19]. In the method of Salvucci et al., human's work is described by rule sets consisted of regular grammars and is defined as a process model. Then, by automatically matching the eye tracking data with the process model, their method estimated the executed process model. In the fixation tracing method, Hidden Markov Model (HMM) [20]

is combined to map to each state, and the whole process model is expressed. By giving fixation points to the obtained HMM, the selected process model is probabilistically calculated.

Based on the knowledge of Salvucci et al., this paper focuses on the transition of fixation from the features of eye movement and processes measured data similar to the retention tracking method. In this paper, the process model corresponds to each line of the program, and we try to construct a thinking model from the transition between processes. First, focusing only on the fixation motion in eye movement, we can obtain the time series of filtered eye positions. To extract the filtered eye data, these paper employes i-vt fixation filter which is a technique to extract only the fixation [21]. To analyze programmer's comprehension process, the transition data from a line to another line is necessary. Therefore, the eye movement is converted to the transition of gazed lines, i.e., fixations are converted based on the range of lines. Figure 2 shows the method to define the row range and the details are as follows. First, in order from the top line, the minimum range of rectangle is determined for each line in which characters are written. Next, the row range is identified by adding the values of pixels corresponding to 1/2 of each line space to the place around the obtained minimum rectangles. This row range enables us to transform the raw data of eye movement into the series of the gazed line.

3.3 Markov Process

Markov process is a stochastic process in which future predicted values are determined only by current observations. In this paper, we use a simple Markov process determined by only the last state among them. By considering a part of source code including a process as state and the transition between parts as edge, we construct the state transition model and regard it as the learner's thinking model. Markov property certainly exists in the transition of eye movement [22]. The target of this paper is only the transition between the lines in source code. Therefore, it is not necessary to emphasize that there is a hidden state between the output symbol string of the observed eye movement and the internal cognitive process. Based on this reason, we adopt simple Markov model, not Hidden Markov model. Also, even in the gaze movement, we focus only on the transition of eyes from one row to another, so we do not think about the transition to the same state. For example, the eye transition according to each line Fig. 3(c) is represented by a simple Markov process as Fig. 4. The edge from node 1 to node 2 in Fig. 4 represents the probability of transition to the second line from the first line in the case when gaze line is staying on the first line.

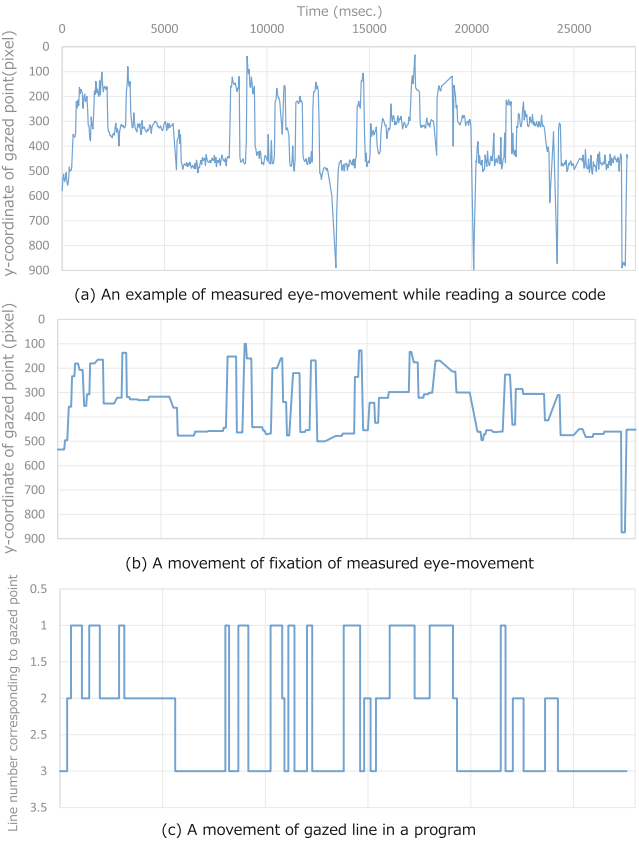


Fig. 3. Flow of processes for measured eye movement

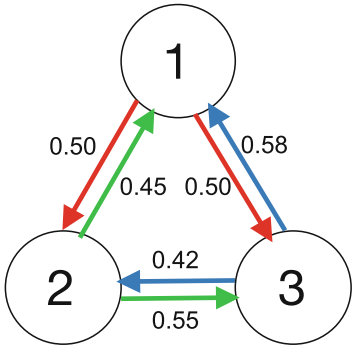


Fig. 4. Simple Markov process of eye movement

4 Experiment

4.1 Protocol

We conducted experiments with examinees who are third, fourth grade, and graduate students majoring informatics and have already learned the basics of programming, such as the foundation of C language, Java language, and the foundation of the algorithm. Regarding the achievement results related to programming, the skill levels of examinee were uniform. This paper conducted experiments in a room where the inside was invisible from the outside to make examinees relaxing. Besides, this paper gave enough consideration to make all examines concentrate on the experimental subject. The first experiment was conducted with 15 people and the second experiment was conducted with 15 people. In the second experiment, we performed on the same 14 people as the first experiment, i.e., one examinee was different from the first experiment. The reason for dividing the experiment into two was to confirm the result of the first experiment and to clarify whether it is a general trend or not by acquiring the data of different subjects. In this paper, we focused on the source code consisted of 3 lines as experiment subjects. All source code was composed only of basic assignment statements and did not include instructions such as increment/decrement and compound assignment operators. The used operators were only arithmetic and surplus operation, and it consisted only of simple statements calculatable in his/her head. In addition, we informed examinees beforehand that all types of variables are integers in advance. There are four kinds of combinations of data dependency relationship in the case of three lines of source code. All source codes were automatically generated for each data dependency relationship and used for experiments. In the first experiment, the examinees addressed 12 subjects (3 source codes for each data dependency). The experimental time was about 10–15 min. In the second experiment, the examinees addressed the 8 subjects (2 source codes for each data dependency). In order to measure the eye movement, X2-30 Eye Tracker produced by Tobi Technology Inc. was used. Examples of the source code used in the experiment is shown in Fig. 5.

Figure 6 shows the experimental situation. The examinee read the presented program and answers the values of all variables after its execution. Each source code was presented in an irregular order, and the examinee was not informed the type of data dependency. We finished gaze measurement at the time when the examinee answered the values of all variables. The timing of answer was free. For example, at the time when the value of only one variable in the presented program is found, it is possible to answer its value. The examinee continued reading until he/she had the correct value for all variables. We paid attention to examinees in advance so as not to respond at a venture. The source code was displayed at the 12.1-in. display with fullscreen. Regarding display setting of the source code, the left margin was 100 px, the upper margin was 40 px, the font size was 50 px, and the line spacing was 150 px, which were empirically determined but the comfortable settings for us. The flow of the experiment was as follows.

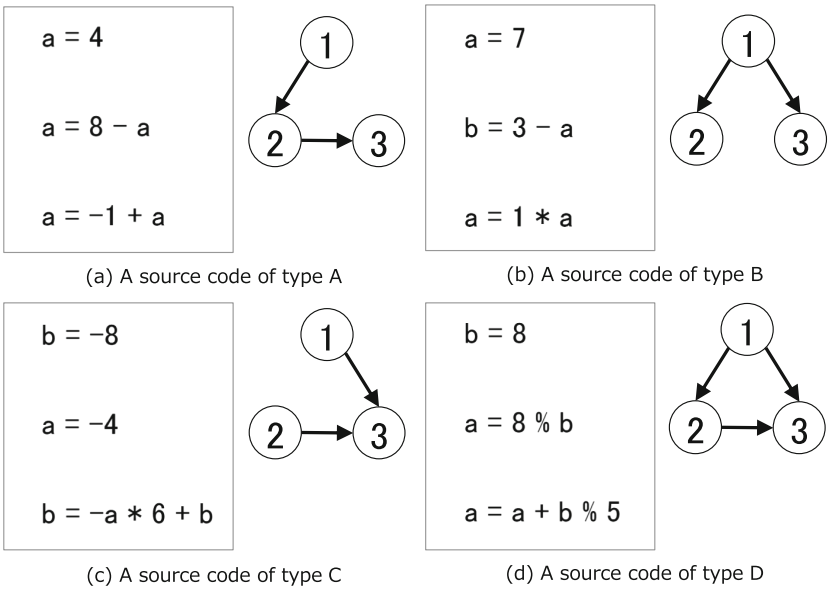


Fig. 5. Source codes used in the experiments

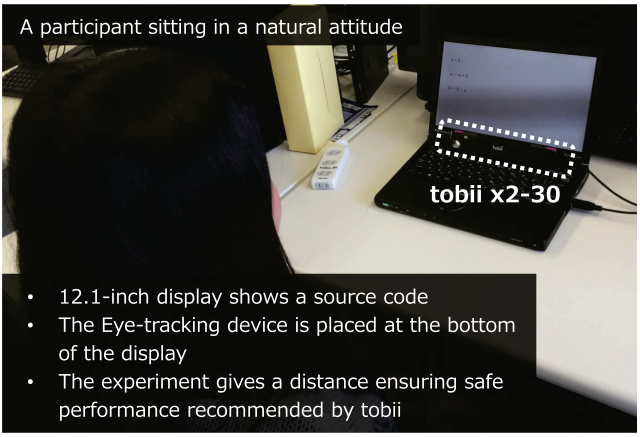


Fig. 6. A picture showing actual condition of the experiment

1. Each examinee was given to explanation about the experimental method such as the presentation method of the subject, the answer method, the experimental time, things to keep in mind when reading source code (do not close your eyes, do not move your face, and keep proper posture). After the confirmation of essential matters, calibration was performed (about one minute).
2. We explained the detail of the subject and do a preliminary exercise of the experiment with the examinee (about two minutes).

3. We measured the data of eye movement one by one. After receiving the examinee's answer, we ended the measurement (about 10–30 s per question).
4. After completing the experiment of one question, we prepared a sufficient time break (about 10–30 s). After the break, went to the next question. We presented the problem in the same way as the above procedure and finished when performing all the tasks.

4.2 Result

Markov models were generated for each reading task from the eye movement data of each examinee. As the targets of this paper were four types of data dependency relationship. This paper calculated the averages of the transition probabilities of Markov model between nodes for each Type as shown in Fig. 7. In Fig. 7, the transition probability from the node i to the node j is represented as $i \rightarrow j$, and the error bar indicates upper and lower limits of the population mean (infinity) with 99% reliability when assuming t-distribution. Also, the probability of each data structure is indicated by a solid line, and the average of the probabilities of all data structures is connected by a broken line. From the result of Fig. 7, it can be confirmed that the features corresponding to the data dependency relationship were reflected the eye movement. We can observe that there are some edges with the large difference from the average, and this pattern of difference corresponds to the data dependency relationship.

The basic reading strategy in all experimental tasks consists of the scanning operation for the whole from the top to the bottom to understand the data

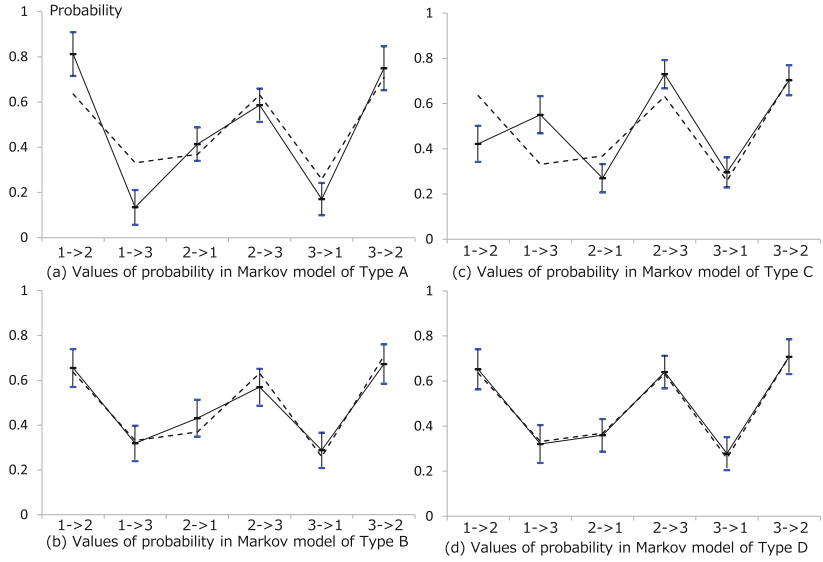


Fig. 7. Average probabilities in Markov model of all participants

dependency relationship, and the reading operation based on the structure after returning to the top. Uwano et al. showed three patterns on the programmer's eye movement: behavior to view the entire code from the top to the bottom immediately after the review started, behavior to check the declaration part of variables when each variable is first referenced, and behavior to check the previous line containing a variable when the variable appears [4,5]. While correlating Uwano's findings and time series of examinee's eye movement, we qualitatively considered the reading strategies of each structure. The basic flow of each structure depended on the order of variables to be identified, and this difference affects the eye movement.

Considering the data structure of Type A, it is supposed that this common reading strategy is based on a flow of understanding while identifying variable values in order from the top after scanning. Therefore, it can be predicted that the transition probabilities of the 1st to the 3rd line and vice versa can be lowered. In the result of Fig. 7, as expected, the transition from the 1st line to the 3rd line and the transition from the 3rd line to the first line, both of which have no data dependence, were lower than these pair transitions. The transition from the 3rd line to the 1st line was significantly lower, and it can be said that it was a tendency strongly influenced by the structure even if considering the regression (the action to return to read the sentence already read).

The common reading strategy of Type B has, firstly, the motion to understand the instructions in order from the top after scanning. Next, since there is a dependency relationship between the 3rd line and the 1st line, there is a motion to read returning to the 3rd line once confirming the 1st line. In other words, it can be predicted that there is the reading motion including the reciprocating motion between the 1st line and the 3rd line in addition to the flow of the reading of Type A. As expected, the shape of the result of Type B was similar to Type A, but the transition probabilities between the 3rd line and the 1st line was increased as compared with Type A.

The common reading strategy of Type C is as follows. After scanning, there is a reading motion to check the 3rd line, move the center of sight to the 3rd line, and repeat minute round-trip with the 1st line and the 3rd line to identify the value of the variables. Therefore, the transition probabilities from the 1st line to the 2nd line and vice versa are considered to be lower than the other three structures. In the result of Fig. 7, as expected, the transition probabilities from the 1st line to the 2nd line and vice versa were lower than the other structures.

In the case of Type D, the value of the 3rd line is determined after the value of the variable is determined in the 2nd line. Therefore, after scanning, the common reading strategy of Type D consists of the motions to repeat minute round-trip with the 1st line and the 2nd line, move the center of sight to the 3rd line, and repeat minute round-trip between the 3rd line and the 1st line and between the 3rd line and the 2nd line. Type D has the most complicated structure among the four patterns of data dependence. That is, it can be predicted that Type D follows a reading strategy that includes all reading motions of other data structures together. In the result of Fig. 7, the overall probability average and

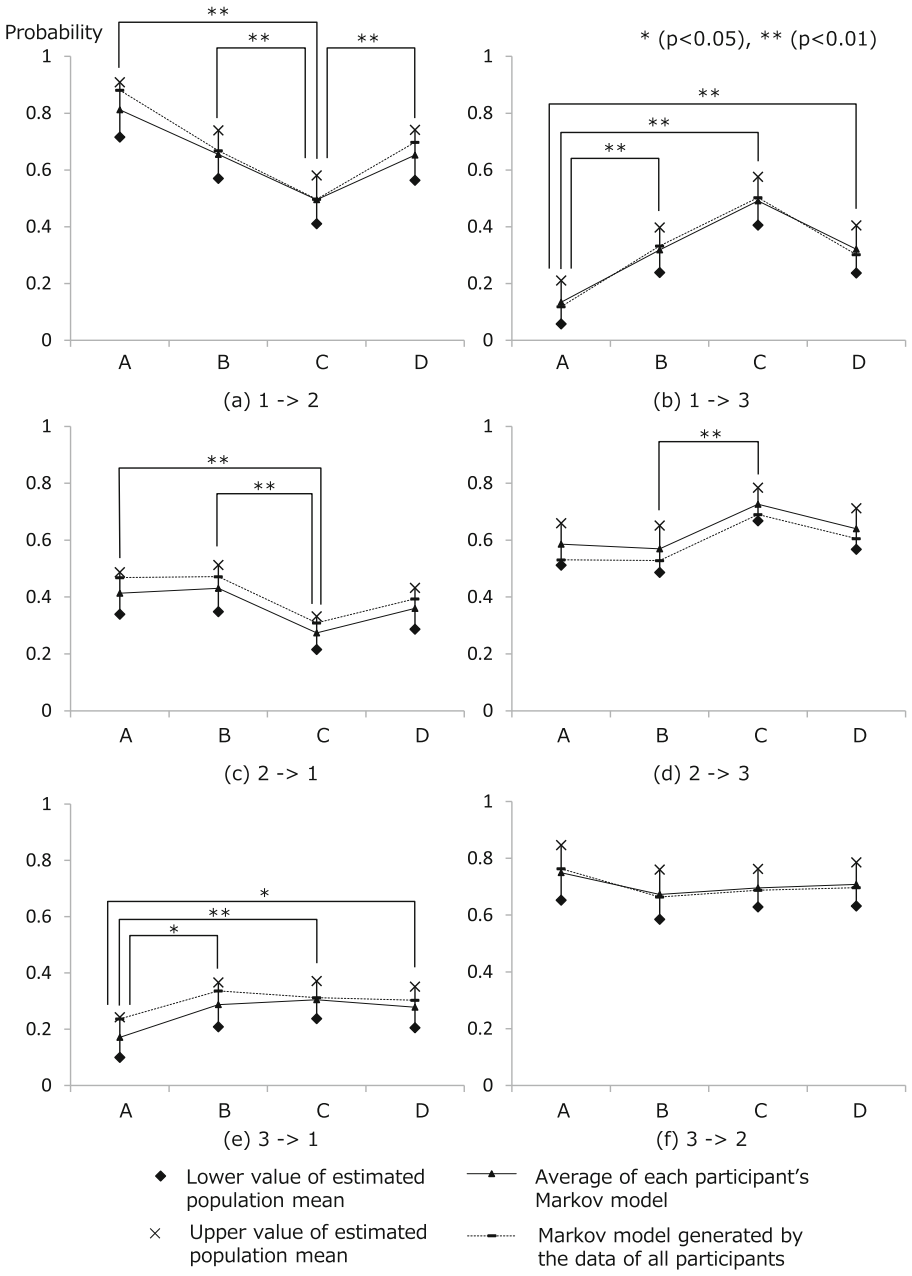


Fig. 8. Comparison of each structure's average probability

Type D probability were almost similar. Therefore, as expected, the reading of Type D was undoubtedly the result of adding together all reading motions of other data structures.

As the overall tendency, the transition probabilities of the 1st line from the 2nd line and the 1st line from the 3rd line were lower than other. The reason might be that the 1st line in many cases was a definition sentence, and it was relatively easy to compare with the content of the 3rd line.

Next, Fig. 8 shows the result of comparing the average of transition probabilities between nodes i, j for each structure. The error bar of Fig. 7 indicates upper and lower limits of the population mean (infinity) with 99% reliability when assuming t-distribution. The probability of each data structure is connected by a solid line, and the probability generated by combining all examinees' eye data is connected by a broken line. The purpose of combining all examinees' data is to examine only the influence of the data dependency relationship from the eye movement without relying on each examinee's feature. From Fig. 8, all the transition probabilities of all examinees were within confidence intervals. Furthermore, the difference of structure certainly appeared in the difference of transition probability. Therefore, we confirmed that general tendency of eye movement according to the structure could be shown by combining all examinees' data when analyzing the difference in the reading pattern according to each structure.

As for the transition probability between nodes 1 and 2 shown in Fig. 8 (a), Type C without dependency relationship between nodes 1 to 2 is significantly lower than other structures by Tukey-Kramer test ($p < 0.01$). Similarly, for the data dependency relationship between nodes 1 and 3 shown in Fig. 8(b), Type A without dependency relationship between nodes 1 and 3 was significantly lower ($p < 0.01$). Also in Fig. 8(c), Type C without dependency relationship between nodes 2 and 1 was significantly lower ($p < 0.01$) than Type A and B, but there was no significant difference with Type D. In Fig. 8(d), Type B without dependency relationship between nodes 2 and 3 had the lowest probability, but there was an only significant difference with Type C ($p < 0.01$). Between nodes 3 and 1 shown in Fig. 8(e), Type A, which has no dependency relationship between nodes 3 and 1, had the lowest probability. Type A was significantly lower than Type B and D ($p < 0.05$), and was also significantly lower than Type C ($p < 0.01$). Between nodes 3 and 2 shown in Fig. 8(f), Type B with no dependency relationship between nodes 3 and 2 had the lowest probability, but there was a significant difference from other structures. From the above, it can be said that the features of each structure appeared in the transition probability of eye movement, and the effectiveness of the proposed modeling method and the usefulness of the eye movement for estimating the reading process were clarified.

As an effort for reading source code, Orlov et al. proposed a method to convert a source code into an abstract semantic element and model an eye movement using HMM [23]. However, Orlov et al. concluded that his achievement is only one guideline because they could not obtain a result clearly supporting the validity of the model. On the other hand, this paper could obtain meaningful results from

the approach similar to Orlov's one. Therefore, the findings of this paper would be valuable for further expansion and application of the Markov model-based method.

5 Conclusion

This paper investigated the possibility of estimating learner's program comprehension process during reading based on the pattern of eye movement, not the eye distribution. From the viewpoint of program slicing, the internal structure of the program was restricted to the data dependency relationship, and the features of eye transition at lines of source codes were clarified. First, we got the eye movement during programming reading process and proposed its modeling method. Then, based on the program structure, we designed an experimental protocol to analyze the eye movement during reading source code. This study constructed programs with only assignment and arithmetic operations, simplified the structure of programs to data dependency relationship, and set them as analysis targets. From the data of such programs obtained by eye tracking, this study constructed a simple Markov process model to express eye movement patterns as transition probabilities. This paper used four kinds of combinations of data dependency relationship in the case of three lines of source code at the experiment. Some source codes were automatically generated for each data dependency relationship and used for experiments. By comparing the difference of transition probabilities between nodes and data dependencies, this study showed the possibility of analyzing the program reading process based on eye movement. As a result, we could confirm the influence of each pattern's data dependency relationship as the peculiar behavior of program reading.

Although the experimental subjects in this paper were simple, we were able to show the tendency of eye movement based on data dependency relationship. Therefore, based on the result of this paper, in the future, we will identify the eye trend of learners who have a difficulty to read a source code. The analysis will be possible to discriminate learners who are out of the criteria, standard eye movement depending on data dependency relationship, and to instruct an efficient reading method. Also, the increase of the size and complexity of program will contribute to elucidating the implicit thinking process which was difficult to clarify by conventional methodology alone.

Acknowledgments. This work was partly supported by Japan Society for the Promotion of Science, KAKENHI Grant-in-Aid for Scientific Research(C), 16K01147, 17K01164.

References

1. Sagisaka, T., Watanabe, S.: Development and evaluation of a web-based diagnostic system for beginners programming course. J. Jpn. Soc. Inf. Syst. Educ. **27**(1), 29–38 (2010). (in Japanese)
2. Pennington, N.: Stimulus structures and mental representations in expert comprehension of computer programs. Cogn. Psychol. **19**, 295–341 (1987)
3. Okamoto, M., Terakawa, K., Murakami, M., Ikeda, K., Mori, M., Uehara, T., Kita, H.: Computer programming course materials for self-learning novices. In: Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications, vol. 2010, no. 1, pp. 2855–2861 (2010)
4. Uwano, H., Nakamura, M., Monden, A., Matsumoto, K.: Exploiting eye movements for evaluating reviewer's performance in software review. IEICE Trans. Fundam. **E90–A**(10), 317–328 (2007)
5. Uwano, H., Nakamura, M., Monden, A., Matsumoto, K.: Analyzing individual performance of source code review using reviewers' eye movement. In: Proceedings of the 2006 Symposium on Eye Tracking Research & Applications, pp. 133–140 (2006)
6. Kashima, T., Matsumoto, S., Yamagishi, S.: Knowledge acquisition with eye-tracking to teach programming appropriate for learner's programming skill. In: Proceedings of the Third Asian Conference on Information Systems, pp. 287–292 (2014)
7. Rayner, K.: Eye movements in reading and information processing: 20 years of research. Psychol. Bull. **124**(3), 372–422 (1998)
8. Ihantola, P.: Notes on eye tracking in programming education. In: Eye Movements in Programming Education, pp. 13–15 (2014)
9. Crosby, M., Stelovsky, J.: How do we read algorithms? A case study. IEEE Comput. **23**(1), 24–35 (1990)
10. Weiser, M.: Programmers use slices when debugging. Commun. ACM **25**(7), 446–452 (1982)
11. Agrawal, H., Horgan, J.: Dynamic program slicing. In: SIGPLAN Notices, vol. 25, no. 6, pp. 246–256 (1990)
12. Nishimatsu, A., Nishie, K., Kusumoto, S., Inoue, K.: An experimental evaluation of program slicing on fault localizaion process. IEICE Trans. Inf. Syst. **82**(11), 1336–1344 (1999). (in Japanese)
13. Ishio, T., Kusumoto, S., Inoue, K.: Debugging support for aspect-oriented program based on program slicing and call graph. In: Proceedings of 20th IEEE International Conference on Software Maintenance, pp. 178–187 (2004)
14. Inazumi, H., Takeuchi, S.: How to learn programming technique by using program slicing. Aoyama Inf. Sci. **29**(1), 51–78 (2001). (in Japanese)
15. Yoshida, H., Tateiwa, Y., Yamamoto, D., Takahashi, N.: A code review navigator with chunking and slicing for assembly programming exercise. IEICE technical report, vol. 109, no. 335, ET2009-81, pp. 169–174 (2009). (in Japanese)
16. Ottenstein, K., Ottenstein, L.: The program dependence graph in a software development environment. In: Proceedings of ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, SDE 1, pp. 177–184 (1984)
17. Rayner, K.: Eye movements in reading and information processing, 20 years of research. Psychol. Bull. **124**(3), 372–422 (1998)

18. Salvucci, D., Anderson, J.: Automated eye-movement protocol analysis. Hum.-Comput. Interact. **16**(1), 39–86 (2001)
19. Ohno, T.: What can be learned from eye movement?: understanding higher cognitive processes from eye movement analysis. Jpn. J. Cogn. Sci. **9**(4), 565–579 (2002). (in Japanese)
20. Juang, B., Rabiner, L.: The segmental k-means algorithm for estimating the parameters of hidden Markov models. IEEE Trans. ASSP **38**(9), 1639–1641 (1990)
21. Olsen, A.: The Tobii I-VT Fixation Filter, Tobii Technology (2012)
22. Iwao, T., Mima, D., Kubo, H., Maejima, A., Morishima, S.: Analysis and synthesis of eye movement in face-to-face conversation based on probability model. Inst. Image Electron. Eng. Jpn. **42**(5), 661–670 (2013). (in Japanese)
23. Orlov, P.: Primary investigation of applying Hidden Markov Models for eye movements in source code reading. In: Eye Movements in Programming Education II: Analyzing the Novice's Gaze, pp. 18–20 (2015)