Vol.2022-HCI-197 No.9
2022/3/14

マルチモーダル情報を用いたプログラミング時のつまずき検出手法の提案

岡 大貴[1,a)]　大西 鮎美[1,b)]　寺田 努[1,c)]　塚本 昌彦[1,d)]

概要：プログラミングにおいて，プログラムの実行結果が想定していた結果と異なる，という状況に直面することは多々ある．この際にエラーメッセージが表示されていれば，プログラムを修正するための手がかりを得られるが，「エラーは出ていないが，実行結果が想定と一致しない」場合，この状況を解決するのは容易ではない．特にプログラミング初学者にとって，このようなつまずきが長時間続くことは，モチベーションの低下や挫折を引き起こす原因となる．本研究では，このつまずき状況をプログラマのプログラム内容と，人の内的状態を表す指標の一つである心拍情報を併用し，自動検出することを目指す．本論文では静止画をプログラム実行で再現するタスクを題材に，つまずきの検出に有用であると考えられるパラメタをいくつか算出し，つまずいている状態を検出できるかを検討した．

1. はじめに

プログラミングを学ぶ過程では，学習者はプログラミング言語の文法や言語仕様，プログラミングにまつわる概念を，実際にプログラムを書くことを通して理解する．大学でのプログラミングに関する講義では，学習内容に関する課題を受講者にプログラミングで解かせることで理解度を把握し，独学におけるプログラミング学習においても，ゲームなどのソフトウェアをプログラムで実装することで学習を進める場合が多い．しかし，プログラミングにつまずきはつきものである．

プログラム実行時にエラーメッセージなどのプログラムを改善するための手がかりがあれば，それをもとにプログラムを修正し，つまずきから復帰できるが，プログラムを実行できておりエラーメッセージは無いが実行結果が想定したものと異なる場合，つまずきから復帰することは難しい．

学習者のプログラム内容からつまずきを検出・支援することを目指した研究はいくつか存在するが [1–3]，プログラム内容のみから学習者の状況を把握することは難しく，リアルタイムにつまずきを検出したり，高い精度でつまずきを検出する実用的なシステムの実装には至っていない．

本研究では，プログラミング学習者の生体情報とプログラム内容を併用することで，学習者のつまずきを検出する手法を提案する．提案手法では，生体情報の中でも計測の負荷が少ないが，学習者の内的状態を推定できる指標である心拍情報を用いる．これにより学習者の集中や心的負荷を考慮しつつ，プログラム内容とそれを照らし合わせることで，高い精度でつまずきを検出できるのではないかと考えた．

今回はプログラミングに関する講義などで課題が出題されており，それを学習者が解くという場面を想定した実験を行い，つまずきの検出を目指した．このような場面でつまずきが検出できれば，教授者に対し学習者のつまずき状況を可視化し，サポートを円滑に行うことや，つまずき時のプログラム内容に即した情報提示などを行うことができると考えられる．

本論文では以降，2 章で関連研究を紹介する．3 章でつまずき検出に用いる指標について説明し，4 章でその指標の変化を観察する実験について述べる．5 章では実験で得た結果からつまずき検出手法を提案し，6 章で実験結果に対する考察と議論を行う．そして 7 章で本論文をまとめる．

2. 関連研究

本章ではプログラミング時のプログラマの状態を把握し，支援することを目的とした先行研究について述べる．

[1] 神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University
[a)] hiroki-oka@stu.kobe-u.ac.jp
[b)] ohnishi@eedept.kobe-u.ac.jp
[c)] tsutomu@eedept.kobe-u.ac.jp
[d)] tuka@kobe-u.ac.jp

2.1 プログラム内容を用いたプログラマの状況把握に関する研究

プログラミング時のプログラム内容を分析することで，プログラマの状況を把握しようとした研究は多く存在する．伏田らは初学者を対象としたプログラミング演習においてソースコードの編集履歴を分析することで，初学者が陥りやすいコーディング時の誤りパタンを観測した [8]．この研究で観測された誤りパタンは 2 つあり，1 つ目は修正すべき箇所とは関係ない箇所を修正し続けるというパタンであり，2 つ目は習得している他のプログラミング言語の知識に引きずられてエラーを引き起こすというパタンであった．また，蜂巣らはプログラミングの演習課題において学習者が編集したプログラムの字句，式，文，文の構造に着目し分析することで，誤ったプログラムの記述や別解を検出している [9].

なお，プログラムを構文解析し，抽象構文木 (AST: Abstract Syntax Tree) を得ることでプログラムを定量的に評価する試みも盛んに行われている．漆原らは授業課題における提出プログラムを AST を用いて採点することを試みている [10]．この研究では，Python を用いたプログラミングの授業課題における提出プログラムから AST を作成し，その AST に課題の想定解であるプログラムの AST と同じ構造木が部分木として含まれているかを比較することで，提出プログラムが正解かを判定し，ある程度妥当な判定が可能であることを示した．

このように，プログラムに関する様々なデータに着目し，学習者の状況を把握する手法がとられているが，プログラム内容のみから学習者の置かれている状況を把握することは難しい．本研究で提案する手法では，プログラム内容と生体情報を併用することで，学習者がつまずいている状況を検出する．

2.2 生体情報を用いたプログラミング時の状態分析に関する研究

プログラミング時の生体情報を取得し，その際のプログラマの活動を分析する研究はいくつか行われている．Siegmund らはプログラム理解時の脳を fMRI を用いて分析し，ワーキングメモリや言語処理などに関連する脳領域が活発になることを明らかにした [4]．また Ishida らはプログラマの脳波と参照しているドキュメントタイプを用いて，プログラムの理解度を把握することを目指した [5]．この研究ではコンピュータサイエンスを専攻している学生に Java のプログラムを読解するタスクを課し，その際の脳波と参照しているドキュメントタイプからタスクの成否を 85.2%の精度で分類することができた．さらに，Crosby らはプログラマの視線を分析し，プログラミング熟練者はプログラム内の重要な箇所や複雑な命令に着目するのに対し，初心者はコメントや比較文に視線が集中するという傾向を明らかにした [6]．Uwano らはソースコードレビューの際，レビュアの眼球運動を計測する DRESREM というシステムを実装した [7]．このシステムを用いてソースコードレビュー時の眼球運動を分析した結果，コード全体を上から下へ眺めた後，細部に注目する scan というパターンがあることを発見し，scan に十分な時間をかけないレビュアはソースコード中の不具合を発見するのに時間がかかっているという結果を得た．

このように脳波や視線といった生体情報から，プログラミングに関する活動を分析しているものは存在するが，脳波や視線を計測するデバイスは高価であったり，装着すると活動を阻害する可能性があり，実際のプログラミングに導入するには課題がある．本研究では，比較的安価で装着時の負担の小さい心拍センサを用いてプログラマの心拍情報を取得し，プログラマの状況を分析する．

2.3 プログラミング時のつまずき検出・支援に関する研究

プログラミング学習において，学習者のつまずきを検出するための研究はいくつか行われている．藤原らは学習者がプログラミング課題に取り組んでいる際のプログラムに関する時系列データを用いて，学習者の行き詰まりなどの特徴を定量化する手法を提案している [1]．この研究では C 言語の課題に取り組んでいる際の学習者のソースコードの行数やコンパイルの回数，エラー数などを分析した結果から，コンパイルの回数とエラーの発生回数という 2 種類のデータの組み合わせが取り組み状況を把握するのに有効であると述べている．浦上らはプログラミングの講義課題において，受講者の課題進捗が滞ることをつまずきと定義し，受講者の提出したプログラムのログを解析することでつまずきの検出を目指した [2]．この研究では，受講者が最終的に完成させた正解のプログラムと，それまでの実行時刻毎のプログラムの差分を比べ，同じである行数を正解行数とし，このパラメタの変化と目視によるつまずきを照らし合わせることで，つまずき時の正解行数の特徴を探っている．また井川らは受講者の課題進捗をロギングするプログラミングツールを用いて，プログラムが自動保存された割合と保存されたが実行に失敗している割合を指標とし，学習者が問題に直面している場面の検出を目指している [3].

しかし，学習者のつまずきの状況は多岐に渡り，それらを正確に検出することは難しいのが現状である．またリアルタイムにつまずきを検出する手法と，つまずき時に学習者をどのように支援してゆくかを具体的に提示した研究は筆者らの知る限り存在しない．

2.4 心拍情報を用いた学習状況把握に関する研究

心拍情報を用いて学習者の活動や状況を把握することを目指した研究はいくつか存在する．永田らは生体情報計測装置を用いて学習時の心拍数や心拍変動を計測する実験

を行い，暗算課題に取り組んでいる被験者は心拍数が増加し，否定的感情が高まるなど，安静感情が抑制された状態となっていることを明らかにした [11]．また，同実験内での音楽聴取課題中では，否定的感情は抑制され，安静感情が高い状態となっていることを発見した．宮西らは指尖容積脈波を用いて大学生の学習活動時の心拍変動を計測した結果，LF/HF 値から算出したストレス指標値と学習者の理解度，疲労度の間に関連があり，理解度と疲労度からストレス指標値が推定できる可能性を示した [12]．

このように，心拍情報は学習状況を把握するための指標として有用である．本研究では，プログラミングにおける学習者の学習状況を把握するために，心拍情報を活用する．

3. つまずき検出に用いる指標

本研究では，プログラマのプログラミング時の心拍情報とプログラムに関する情報を併用することで，つまずきを検出することを目指す．以下で，本研究で対象とするつまずき状況とその検出のために用いる指標について説明する．

3.1 対象とするつまずき状況

本研究ではプログラミングに関する講義の受講者が課題に取り組んでいる状況を想定し，その際のつまずきを検出する．

ここでいうつまずきとは，「プログラマの想定しているプログラムの実行結果と，実際のプログラムの実行結果が乖離している状態が長時間続くこと」である．プログラムを実行した際にエラーが出力された状況をつまずきとみなす考え方もあるが，「エラーは出ていないが，実行結果が想定と一致しない」という状況もプログラマのつまずきであると考えられる．すなわち，エラーが出ている箇所はプログラマがつまずいている可能性が高いが，そうでない箇所でもつまずいている可能性があり，エラー情報のみではつまずきを検出するには不十分であると考える．

このつまずきを検出するため，本研究では心拍情報を用いることで，プログラマの精神的負荷や集中・非集中などの状況を考慮し，正確なつまずき検出ができるのではないかと考えた．

また，講義など教授者が学習者にプログラミングを教えている状況であれば，学習者のつまずきを検出した際に教授者にそれを通知し，円滑な支援を実現することができる．よってリアルタイムにデータ変化を観察し，つまずきを検出，学習者にフィードバックを行うといった支援システムを想定したつまずき検出手法を検討する．

3.2 具体的な指標

本論文では，つまずき検出のため以下の 4 つの指標を選出した．

(1) **LF/HF 値**

一般に，心拍のピークである R 波と次の拍動における R 波の間隔 (RRI: R-R Interval) の変動に，交感神経活動および副交感神経活動の変化が反映される．RRI の変動を周波数解析した際の，低周波成分と高周波成分の比率を LF/HF 値といい，これは交感神経と副交感神経の全体のバランスを表す．本研究ではこの LF/HF 値の大きさを，集中度や精神的負荷を反映する尺度として，学習状況の把握に用いる．

(2) **プログラムの実行回数・実行間隔**

学習者がつまずいている際は，プログラムの実行回数が少なくなったり，実行間隔が大きくなると考えられる．本研究では学習者のプログラムの実行間隔の変化をつまずきの 1 つの指標として観察する．

(3) **プログラム内容**

プログラミング時の学習者の学習状況は，プログラムの行数や変更差分などに表れると考えられる．本研究では，学習者のプログラム実行毎にプログラムの行数の変化を観察する．

(4) **エラー内容**

プログラム実行時のエラー内容やエラーメッセージは学習者のつまずきに密接に関連している．実行時のエラーが連続している箇所は学習者がなぜプログラムが誤っているのかを理解できず，つまずいている可能性がある．本研究では学習者がプログラミングしている際の実行毎のエラー内容を収集し観察する．

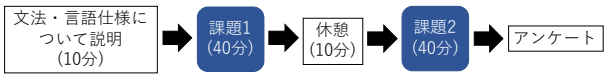

図 1 実験の流れ

4. 実験内容

3 章で説明した指標がプログラミング学習におけるつまずき時にどのように変化するかを観察するための実験を行った．本章ではその実験内容と結果について述べる．

4.1 実験概要

プログラミングの授業などでの課題を解いている環境を想定し，被験者にプログラミングで解く課題を 2 題出題する．その際の学習者の心拍情報とプログラム情報を収集し，分析することでつまずき時のデータ変化を観察する．そして学習者がつまずいていると考えられる箇所を第一著者が目視でラベリングし，先述の指標から検出できるかを検証する．また，使用するプログラミング言語には JavaScript を指定し，描画ライブラリである p5.js [13] を使用させた．

実験の流れを図 1 に示す．実験時には，最初に課題を再現するために必要な JavaScript の文法と p5.js のメソッドについて被験者に約 10 分ほど説明を行った．各課題に取

り組む時間は約 40 分間とし，1 つ目の課題の後に約 10 分間の休憩を設けた後，2 つ目の課題に取り組ませた．なお課題中は Web ブラウザでの検索やメモなどを許可し，自由に課題に取り組ませた．また課題の難易度やつまずき内容について調査するため，実験後にアンケート調査を行った．被験者は工学系の男子学生 6 名 (うち 2 名は他のプログラミング言語の経験者) である．

4.2 課題設計

本実験では，昨今初学者向けのプログラミング講義において取り入れられ始めている，視覚的なフィードバックを利用したプログラミング学習を想定し，静止画をプログラミングで再現する課題を設計した．このような講義で用いるプログラミング言語としてあげられるのが Processing [14] であり，Processing を用いた講義における実践報告は多くなされている [15–17]．本研究では，IDE などを用意せずとも Web ブラウザから実行することができる Processing の JavaScript への移植版ライブラリである p5.js を使用し，データを収集した．

課題で用いた 2 つの静止画を図 2, 3 に示す．課題 1 はチェスボードをモチーフにしたものであり，条件分岐や繰り返しを用いることで比較的簡単に再現できる課題として用意した．課題 2 はシェルピンスキーのギャスケットであり，再帰や一次元セルオートマトンによる再現が考えられるが，一見するだけでは再現することが難しく，つまずきの状態に陥ることを想定して用意した．

4.3 データ収集

本実験では被験者の心拍情報とプログラム情報を収集する．心拍情報の収集には心拍センサの WHS-3 [18] を使用した．またプログラム情報に関しては，図 4 に示すデータロギング用 Web エディタを実装し，使用した．このエディタは上部に実行や実行停止ボタンを備えており，クリックすることでプログラムの実行・停止が可能である．左にはテキストエディタがあり，右に実行結果を表示する領域がある．下部には擬似的なコンソールがあり，JavaScript の Console API の内容やエラーが出力される．またテキストエディタ部分には，構文ハイライトや構文チェック機能を備えた Ace エディタ [19] を用いた．

なお被験者のつまずき状況を目視でラベリングするため，課題に取り組んでいる被験者の顔映像とスクリーン映像を録画した．そして，被験者の課題進捗の指標として，課題の静止画と被験者のプログラム実行毎における出力画像の類似度を用いた．類似度を算出する手法には，画像のずれや回転に強い特徴量マッチングによる比較手法の 1 つである Akaze [20] を用いた．

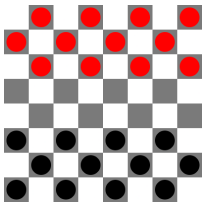

図 2 課題 1

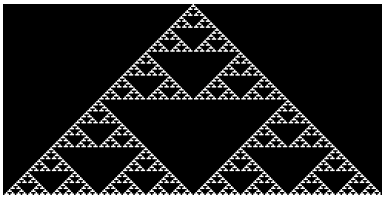

図 3 課題 2

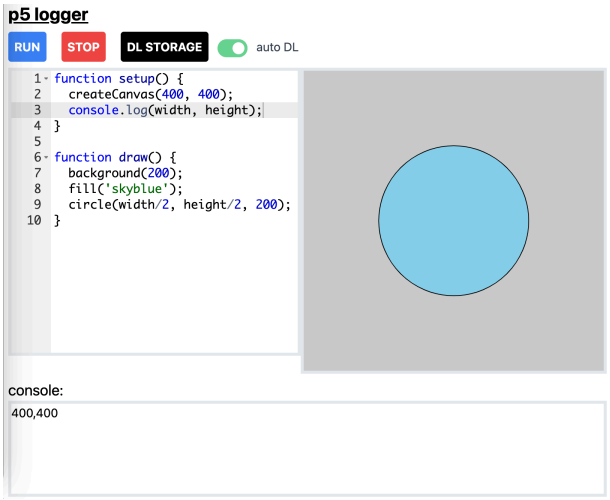

図 4 データロギング用 Web エディタ

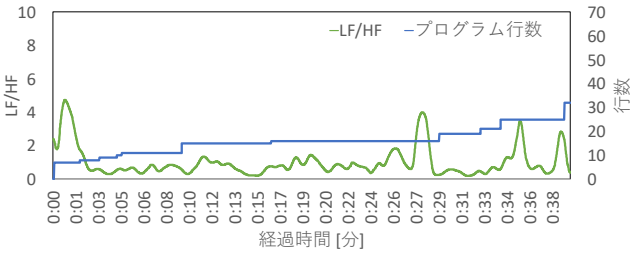

図 5 課題 2 を再現できなかった被験者のデータ変化 (被験者 B)

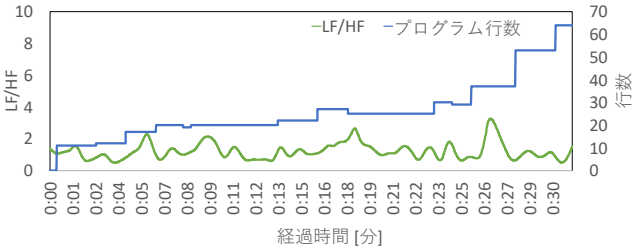

図 6 課題 2 を再現できた被験者のデータ変化 (被験者 E)

4.4 実験結果

本節では実験で得られたデータ変化の観察結果について述べる．

全体の傾向

まず，データ変化の例として，図 5, 6 に課題 2 における被験者 B,E の LF/HF とプログラム行数の変化を示す．被験者 B,E とも課題中に LF/HF 値が変化した箇所があることが確認でき，プログラム行数の変化からプログラムの細かな変更を行っているであろう箇所と大きな変更を行って

表 1 実験結果の概要

	課題 1				課題 2			
	再現	実行回数 [回]	エラー数 [回]	難易度	再現	実行回数 [回]	エラー数 [回]	難易度
A	○	100	-	3	×	53	0	5
B	○	36	4	2	×	27	0	4
C	○	49	1	3	○	56	0	2
D	○	42	1	2	×	17	1	5
E	○	35	1	1	○	43	1	3
F	○	25	0	3	○	20	1	4

表 2 LF/HF 値の比較

	通常時		つまずき時	
	平均	分散	平均	分散
B	0.93	0.72	1.18	0.73
D	2.05	1.17	2.25	1.85
E	1.15	0.28	1.34	0.16

表 3 実行間隔の比較

	通常時		つまずき時	
	平均 [秒]	分散	平均 [秒]	分散
B	97.9	9908.9	178.2	23250.1
C	94.6	8466.8	45.7	2857.4
D	66.4	3093.6	171.7	28787.9
E	86.1	4600.9	128.2	10718.1

表 4 変更行数の比較

	通常時		つまずき時	
	平均 [行]	分散	平均 [行]	分散
B	1.35	3.80	2.16	6.80
C	1.17	5.66	0.25	0.43
D	2.00	10.8	2.00	7.25
E	3.60	27.0	2.00	7.60

いると思われる箇所が確認できる．

また被験者が各課題を再現できたかどうかと，課題中のプログラム実行回数とエラー数，各課題に対する難易度のアンケート結果 (1:簡単, 5:難しい) についてをまとめたものを表 1 に示す．被験者 A の課題 1 に関してはプログラム内容が取得できておらず，エラー数を算出することができなかった．課題 1 は被験者全員が再現できており，課題 2 は半数が再現できていた．難易度に関する評価は，相対的に課題 1 が簡単で，課題 2 が難しいと評価される傾向にあり，期待通りであったが，課題 2 に関して再現できている被験者が想定より多い結果となった．また課題 2 を再現できなかった被験者は課題 1 と比べ，実行回数が減少していた．

各指標の変化

3 章で選出した各指標の変化について，つまずき箇所とそうでない箇所のデータを比較する．課題 2 での実験時の映像を観察した結果，被験者 B,C,D,E においてつまずきと思われる箇所がみられた．よって 4 名の被験者の通常時とつまずき時の各指標の変化について述べる．

(1) **LF/HF 値**

LF/HF 値の平均・分散を比較した結果を表 2 に示す．被験者 C に関しては実験中に心拍センサの電極が外れ，正常なデータが取得できていなかったため，被験者 B,D,E のデータについて述べる．各被験者の課題中の LF/HF 値の変化の傾向として，LF/HF 値が大きく変化する箇所はみられなかったが，通常時とつまずき時の平均値を比較すると，つまずき時においてやや LF/HF 値が大きくなる傾向がみられた．

(2) **プログラムの実行回数・実行間隔**

被験者のプログラム実行間隔の平均・分散を比較した結果を表 3 に示す．全体として，つまずき箇所ではプログラムの実行間隔が長くなる傾向がみられたが，被験者 C に関しては逆に短くなっていた．この被験者 C のつまずき箇所について観察したところ，ある関数の引数を微調整し何度も実行している様子が確認できた．そのため，学習者がアルゴリズムの設計などの構造的な部分でつまずいているのか，細かなパラメタ設定でつまずいているのかなど，具体的にどのような内容でつまずいているのかを把握することが必要であると考えらえる．

(3) **プログラム内容**

被験者のプログラムの変更行数の平均・分散を比較した結果を表 4 に示す．変更行数の変化の傾向は被験者によって異なり，つまずき時の特徴的な変化はみられなかった．行数の変化をみるだけでは，プログラム内のパラメタのみが変更された場合などを検知できないため，より小さな粒度でプログラム内容の差分を観察することや，AST の差分を観察しプログラム構造の変化などを観察することが必要であると考えられる．

(4) **エラー内容**

今回の実験環境ではエラーの発生はあまりみられず，観察された全てのエラーが構文エラーであった．またエラーが長期的に続き，つまずいているといった様子もみられなかった．その要因として，今回の実験ではエディタに構文チェック機能が備わっていたため，被験者が実行前に構文のおかしい箇所を修正できたことがあげられる．

アンケート結果

実験後に被験者に回答させたアンケートの結果について述べる．課題 1 に関するアンケートでは，被験者は図形をプログラムで再現することや JavaScript でプログラムを書くことに慣れていなかったため，for 文で図形を記述することや JavaScript の仕様について理解することに時間がかかった，という意見が得られた．

また課題 2 では，「図形の配置に規則性を感じつつもそれを実際にコードで表現できなかった」や「図形の法則性をプログラムでどう表現すればよいか思いつくのに時間がかかった」といった，課題画像の図形の規則性を考えたり，その規則をどうプログラムで表現すればよいかが分からなかったという意見が多く得られた．

今回の実験全体についての感想を自由記述させたところ，「課題 2 は難しかったが，プログラミングは嫌いではないので，挑戦している感じで基本楽しかった」「図形を配置してパズルみたいな感覚でプログラミングできるのは楽しい」「達成感があって楽しかった」といった楽しさに関するコメントが多く得られた．また，課題時に 40 分という制限時間を設けたことが緊張感を生んでいたというコメントがあった．

5. つまずき検出手法

4 章で述べた実験結果から, 学習者のつまずき時に LF/HF 値が大きくなることやプログラム実行回数の減少，実行間隔が広くなる可能性が示唆された．よって，これらの指標に閾値を設け，LF/HF 値が上昇しかつプログラム実行頻度が低下している状況をつまずき状態として検出する手法を提案する．

今後はこの手法を用いてつまずきを自動検出するシステムを構築し，その推定制度を評価する実験を行う．

6. 議論と今後

本論文で述べた実験内容とその結果について考察する．本研究の実験では，学習者に課題を与え，それを解いている際の学習者の顔映像とスクリーンの映像の録画を観察することでつまずき状況を主観的にラベリングした．しかし，これが全てのつまずき状況を正確にラベリングできていたかは検討の余地がある．つまずき状況のラベリングには，出力画像の類似度のほかに，被験者のタイピングの停滞や，「顔を画面に近づける」「上を向く」「顔に手をあてる」といった表出行動を目安に用いた．被験者のつまずきに関連した表出行動を映像から認識し，タイピングの停滞をキー入力から検出することでつまずき状況をより正確にラベリングできる可能性がある．

また，今回は各実行時刻におけるプログラム実行結果の画像と正解画像を Akaze を用いて比較したが，背景色と描画図形の色が近い場合はうまく類似度を算出できないという問題があった．なお画像の比較では，段階的に図形を完成させていくプログラムでなければ，進捗状況を把握できないという課題も見つかった．よって同様の課題設定における進捗把握の際には，別の画像比較手法や別のパラメタを参照するといった改善が必要である．

さらに，今回のようなプログラムで画像を再現する課題におけるデータ変化が，他の課題設定においても同様に起こるのかを検証する必要がある．

また，今回の実験結果から得たつまずき検出のために有用と考えられる各指標の変化を，どのように併用すればつまずきを検出できるのかを検討する必要がある．単純に各指標に閾値を設けてつまずきを検出する手法や，複数の指標をパラメタとして用いてつまずき状況を定式化する手法などが考えられるが，どのような手法が適切かを今後検証していく予定である．

7. まとめ

本論文では，プログラミング学習者の心拍情報とプログラム内容を併用し，学習者のつまずきを検出する手法を提案した．つまずき検出に有用であると考えられる指標を 4 つ選出し，各指標が学習者のつまずき箇所とそうでない箇所でどのように変化するかを観察する実験を行った．その結果，つまずき時は LF/HF 値がやや上昇し，実行回数が減少し，実行間隔が大きくなる可能性が示唆された．

今後は提案手法を用いてつまずきを自動検出できるシステムを構築し，その推定制度を評価する実験を行う．また，プログラム実行毎の AST の編集距離など，学習者のプログラム情報の変化を反映した指標を導入し，プログラム内容を考慮したつまずき検出手法を検討する．

謝辞 本研究の一部は，JST CREST(JPMJCR18A3) の支援によるものである．ここに記して謝意を表す．

参考文献

[1] 藤原理也, 田口 浩, 島田幸廣, 高田秀志, 島川博光: ストリームデータによる学習者のプログラミング状況把握, 電子情報通信学会第 18 回データ工学ワークショップ (DEWS 2007) 論文集, (June 2007).
[2] 浦上 理, 長島和平, 並木美太郎, 兼宗 進, 長 慎也: プログラミング学習者のつまずきの自動検出, 情報処理学会研究報告, Vol. 2020-CE-154, No. 4, pp. 1–8 (Mar. 2020).
[3] 井川一渓, 谷口雄太, 峰松 翼, 大久保文哉, 島田敬士: プログラミングログ分析による支援の必要な学生の検知指標の提案とフィードバックツールの開発, Vol. 2021-CE-162, No. 14, pp. 1–7 (Nov. 2021).
[4] J. Siegmund, C. Kästner, S. Apel, C. Parnin, A. Bethmann, T. Leich,G. Saake and A. Brechmann: Understanding Understanding Source Code with Functional Magnetic Resonance Imaging, *Proc. of the 36th International Conference on Software Engineering (ICSE 2014)*, pp. 378–389 (May 2014).
[5] T. Ishida, H. Uwano, and Y. Ikutani: Combining Biometric Data with Focused Document Types Classifies a Success of Program Comprehension, *Proc. of the 28th*

International Conference on Program Comprehension (ICPC 2020), pp. 366—370 (July 2020).
[6] M. E. Crosby and J. Stelovsky: How Do We Read Algorithms? A Case Study, *Computer*, Vol. 23, No. 1, pp. 25–35 (Jan. 1990).
[7] H. Uwano, M. Nakamura, A. Monden, and K. Matsumoto: Exploiting Eye Movements for Evaluating Reviewer's Performance in Software Review, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 90, No. 10, pp. 2290–2300 (Oct. 2007).
[8] 伏田享平, 玉田春昭, 井垣 宏, 藤原賢二, 吉田則裕: プログラミング演習における初学者を対象としたコーディング傾向の分析, 電子情報通信学会技術研究報告, Vol. 111, No. 481, pp. 67–72 (Mar. 2012).
[9] 蜂巣吉成, 吉田 敦, 阿草清滋: プログラミング演習におけるコーディング状況把握方法の考察, 研究報告コンピュータと教育 (CE), Vol. 2014, No. 3, pp. 1–8 (June 2014).
[10] 漆原宏丞, 本多佑希, 岸本有生, 兼宗 進: 抽象構文木を利用したプログラミング理解度採点の試み, 研究報告教育学習支援情報システム (CLE), Vol. 2021, No. 16, pp.1–6 (Dec. 2021).
[11] 永田悠人, 長野祐一郎, 森田裕介: 生体情報計測装置を用いた暗算課題遂行時の心拍計測と主観評価の関連, 日本教育工学会論文誌, Vol. 44, pp. 189–192 (Oct. 2020).
[12] 宮西祐香子, 長濱 澄, 森田裕介: 指尖容積脈波計測装置による学習活動時のストレス測定と主観評価の関連分析, 日本教育工学会論文誌, Vol. 41, pp. 149–152 (Nov. 2017).
[13] p5.js, `https://p5js.org/`.
[14] Processing, `https://processing.org/`.
[15] 三好きよみ: Processing による初学者向けプログラミング教育の実践, 情報教育シンポジウム論文集, Vol. 2020, pp. 89–95 (Dec. 2020).
[16] 菊池 誠: プログラミング, 何をどう教えているか: Processing によるプログラミング教育, 会誌「情報処理」, Vol. 52, No. 2, pp. 213–215 (Feb. 2011).
[17] 土肥紳一: Processing によるオブジェクト指向プログラミング入門教育の実践, 情報教育シンポジウム論文集, Vol. 2018, No. 19, pp. 134–141 (Aug. 2018).
[18] 心拍センサ WHS-3, `https://www.uniontool-mybeat.com/SHOP/8600085.html`.
[19] Ace, `https://ace.c9.io/`.
[20] P. F. Alcantarilla and T. Solutions: Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces, *IEEE Trans. Patt. Anal. Mach. Intell*, Vol. 34, No. 7, pp. 1281–1298 (2011).