令和 6 年度　修士論文

プログラム読解における
視線運動のクラスタリング
Clustering of Eye Movements
in Program Reading

大阪公立大学大学院
情報学研究科　基幹情報学専攻
学籍番号　BGA23116
明石　拓也

2025 年 3 月 8 日

概要

情報活用能力が必要となった近年において，初等教育からプログラミングが必修化されるなどプログラミング教育の機会は増加している．一方で，プログラミングに長けた指導者の不足が問題視されている．そのため，プログラミング学習の支援となるシステムの重要性が高まっている．学習システム開発のためには，プログラミングを理解している人の読解方法の傾向をつかむことが重要となっている．

これまで，ソースコード読解時の視線運動を対象とした検証が行われている．被験者にソースコード読解を必要とするタスクを課し，読解時の視線運動をアイトラッカーでの計測によりディスプレイ画面上の視線座標という形で取得し，タスクの正誤との関係を調べる分析が主流となっている．また，過去の研究においてプログラミングを理解している人と理解していない人の間にソースコード内の注視場所に違いがあることが示されている．一方，これまでタスクに利用されてきたソースコードは，変数代入や四則演算のみで構成されるもの，条件分岐や繰り返し文を含むものであるが，いずれも単一のクラスを用いた手続き型のソースコードで，数行から数十行程度の短いものである．つまり，既存の研究では，クラスオブジェクト生成やポリモーフィズムなど，オブジェクト指向の概念を取り入れたソースコードでの検証はなされていない．

本論文では，6 つのクラスで構成され，オブジェクト生成やメソッド呼び出しを含む百行以上にわたるソースコードを用いて同様の実験を行い，クラス単位という従来の研究よりマクロな視点での分析を行う．また，目標達成のため，エディタでのスクロールを必要とする程度に長いソースコードと視線座標とを対応させるためのエディタのプラグインを開発する．分析結果の検証のため，被験者ごとの 6 つの各クラスへの注視時間割合を求め，視線運動の傾向を表す 6 次元のパラメータとする．この傾向を可視化するため，3 次元グラフを用いる．6 次元データを 3 次元グラフで可視化するため，主成分分析で次元圧縮をする．

結果として，オブジェクト指向を取り入れたタスクにおいて，正答者の視線運動は 3 次元グラフ上のある一定の範囲に固まる傾向があり，不正答者のものはよりばらつきが大きいことが分かった．主成分分析での次元圧縮の際に算出した累積寄与率は 90%以上と高く，圧縮前のパラメータでもおおむね同様の傾向であるといえる．このことから，オブジェクト指向を取り入れたソースコードを用いたタスクを課すことにより，読解者がコードを理解していない可能性を検知できることが示唆された．

目次

第 1 部　はじめに

1.1　本研究の背景

情報活用能力が必要となった近年において，プログラミング教育の機会は増加している． 日本では，2020 年度から小学校でプログラミング教育が必修化された [1]． また，令和 7 年より大学入試共通テストでも情報 I が必修化され，プログラミングに関する問題が出題されている [2].

一方で，プログラミングに長けた指導者の不足が問題視されている．そのため，プログラミング学習の支援となるシステムの重要性が高まっている． 学習システム開発のためには，プログラミングを理解している人の読解方法の傾向をつかむことが重要となっている．

これまで，ソースコード読解時の視線運動を対象とした検証が行われている [3][4][5]. 現在の研究では，被験者にソースコード読解を必要とするタスクを課し，読解時の視線運動をアイトラッカーでの計測によりディスプレイ画面上の視線座標という形で取得し，タスクの正誤との関係を調べる分析が主流となっている． これまでタスクに利用されてきたソースコードは，変数代入や四則演算のみで構成されるもの，条件分岐や繰り返し文を含むものであるが， いずれも単一のクラスを用いた手続き型のソースコードで，数行から十数行程度の短いものである．既存の研究では，クラスオブジェクト生成やポリモーフィズムなど， オブジェクト指向の概念を取り入れたソースコードでの検証はなされていない.

1.2　本研究の目的

本研究は，短い手続き型ソースコードにおいて読解者のプログラミング理解度によって視線運動に差異が発生するという知見のもと， オブジェクト指向を含む長大なソースコードにおいて読解者のプログラミング理解度による視線運動に差異が生じるか否かを明らかにすることを目的とする．

1.3　本研究で用いる手段

この目的のため，6 つのクラスで構成され，オブジェクト生成やメソッド呼び出しを含む百行以上にわたる比較的長大なソースコードを用いて実験を行い， クラス単位という従来よりマクロな視点での分析を行う．分析結果の検証のため，被験者ごとの各クラスへの注視時間割合を求め，視線運動の傾向を表すパラメータとする． 傾向の可視化のため，3 次元グラフと 2 次元グラフを用いる．高次元データを 3 次元，2 次元のグラフで可視化するため，主成分分析で次元圧縮をする．

1.4　本研究の結果

上記の分析を行った結果，オブジェクト指向を取り入れたタスクにおいて，正答者の視線運動は 3 次元グラフ上のある一定の範囲に固まる傾向があり， 不正答者のものはよりばらつきが大きいことが分かった．主成分分析での次元圧縮の際に算出した累積寄与率は90%以上と高く， 圧縮前のパラメータでもおおむね同様の傾向であるといえる．このことから，オブジェクト指向を取り入れたソースコードを用いたタスクにおいてもプログラミング理解度に応じて視線運動に差異が生じていることが分かった．

次に，分布の違いを利用して未知データの分類が可能といる意見について考察する．今回用いたクラスごとの注視時間割合ベクトルのデータでは，上述の通り正解者と不正解者で範囲がきれいに分かれているわけではなく，正解者が一定の範囲に集まり，不正解者がより広い範囲に分散するという結果となった． そのため，単純な K-means 法や K-medois 法などのクラスタリング手法では分類が困難であるといえる．

今後は，より多くのデータを集めて正答者の視線運動の傾向をつかみ， その傾向から外れている場合に読解者がコードを理解していない可能性を検知できないか検証することが求められる．

1.5　本論文の構成

本論文では，以下の構成に従って研究成果の報告を行う．
第２章では本研究に際し採用した手法の説明をする．
第３章では本研究のため行った実験の説明をする．
第４章では実験で得られたデータを加工し，分析する過程とその結果を説明する．
最後に，第５章で本研究のまとめを行う．

第 2 部　採用手法

本章では，視線測定に使用するアイトラッカー，データ測定時の課題とその解決策の順で説明する．

2.1　アイトラッカーについて

アイトラッカーとは，ディスプレイを見ている被験者の注視点を画面上の座標として取得できる機器である．このアイトラッカーを用いた注視点座標取得の操作をアイトラッキングと呼ぶ． 主に，ディスプレイの画面に固定して使用するスクリーンベースタイプと，ゴーグル型で顔に装着して使用するウェアラブルタイプが存在する．本研究では，Tobii 社が提供するスクリーンベースタイプのアイトラッカーである，Tobii Pro Spark[6] と Tobii Nano Pro[7] を使用する．

アイトラッカーの仕組みについて，Tobii 社は次のように解説している [8]． 角膜上に光の反射点を生じさせ，その画像をアイトラッカーの内蔵カメラで撮影する．次に，撮影された眼球画像から角膜上の光の反射点と瞳孔を識別し，それらの情報をもとに眼球の方向を算出する．

この 2 つの機器を用いた測定時は被験者の顔の位置・角度を適切な位置に固定する必要がある．また，使用する際にはキャリブレーションと呼ばれる操作を行い，アイトラッキングの精度を保つ必要がある． さらに，設置位置など，アイトラッカーのパラメータを設定する必要もある．これら 3 つの設定は，Tobii 社が提供する Tobii Pro Eye Tracker Manager[9] というソフトウェアを用いて実現できる．

図 1 に，実験に使用した Tobii Pro Nano の様子を示す．図 1 中の赤い四角に示す機器が，ディスプレイ下部に設置された Tobii Pro Nano である． 本機器を起動した状態で被験者にディスプレイ画面上の情報を目で追わせ，読解時の各時点における注視点座標をリアルタイムで取得できる． Tobii Pro Spark についても同様である．

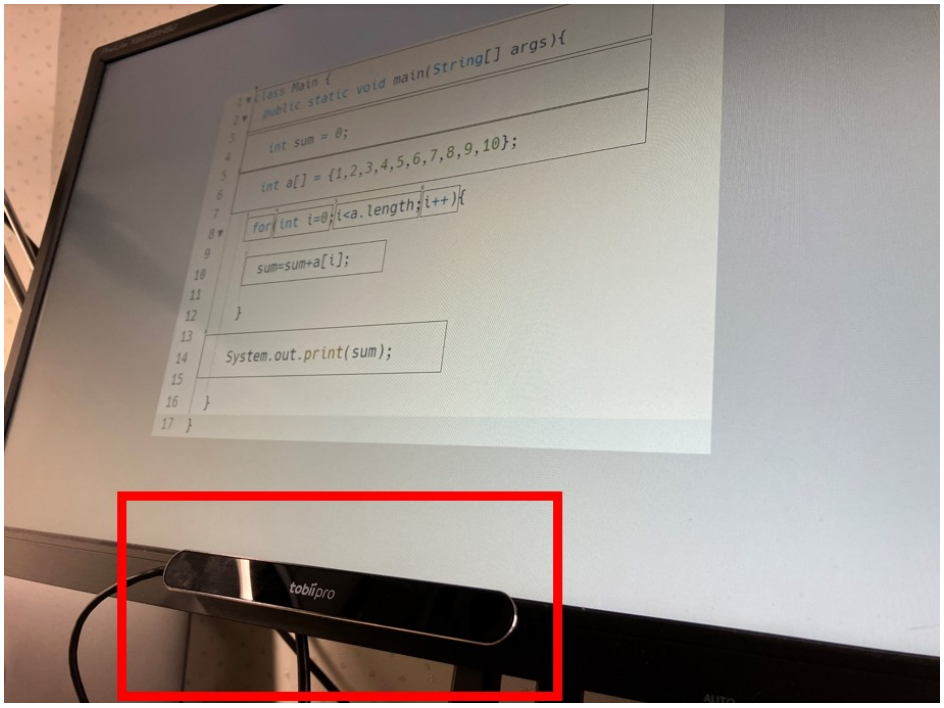

図 1: Tobii Pro Nano

2.1.1 Tobii Pro Eye Tracker Manager

本研究では，Tobii Pro Eye Tracker Manager[9] というソフトウェアを用いてアイトラッカーのパラメータ設定やキャリブレーションを行う． 設定可能なパラメータとして，ディスプレイ画面に対するアイトラッカーの設置位置や角度などがある．

また，本ソフトを用いたキャリブレーションでは，被験者の顔の適切な位置を画面で確認しながら調節できる．

2.2 先行研究の分析手法

本研究では，ソースコードを読解する際の視線座標とソースコード中の要素を対応付ける必要があるが，単純な座標に基づく対応付けではエディタ画面のスクロールが行われた際に対応付けが破綻する問題がある． しかし，長大なソースコードを使用する都合上，スクロール操作には対応する必要がある． この問題の解決のために使用する手法の紹介をする．

吉岡らは，アイトラッカーで取得した視線座標からソースコード中の行・列に変換し，さらに構文木に変換するマッピング手法を提案した [3]． 変換の手順は以下のとおりである．

1. 視線計測装置が被験者の各時点における注視点をディスプレイ上の座標 (例, X:121,Y:313) として時系列に出力する．
2. 座標-行/列変換モジュールが座標単位の視線移動とソースコードを入力として受け取り，ソースコード名と行/列の組 (例，Main.java, 行：1, 列:13) として出力する．
3. このとき，得られた行・列番号からソースコード中の単語を抽出し，構文解析で得られた構文木上のノードと対応をとる．

吉岡らは，このマッピング手法を用いて視線座標をソースコード中の変数名や if,else などの単語単位でマッピングしている． その後，各単語毎の注視時間割合を求め，正解・不正解ごとの割合の差異を確認している．

2.3　本研究の分析手法

本研究では，吉岡らが提案したマッピング手法を使用し，アイトラッカー出力の視線座標をソースコード中の要素と対応付ける． その後,

2.4　iTrace について

本研究では，前章で述べた視線座標とソースコード中の行・列の対応の実装に，iTrace[10] を用いる．iTrace はアイトラッカーでの測定を支援するオープンソースのソフトウェア群で，公式サイトおよび GitHub で公開されている． 主に，アイトラッカーの制御・出力フローの処理等に用いる iTrace Core と，iTrace Core の出力を受け取り，ソースコードを表示しているエディタの情報と結びつけられるプラグインが用いられる．

本研究では，iTrace Core とエディタのプラグインとして iTrace Atom を使用する．以下に，この 2 つの詳細を示す.

2.4.1　iTrace Core

iTrace Core は，アイトラッカーの制御や出力フローの処理に使用できる． 本研究では，アイトラッカーの測定開始・終了操作と，アイトラッカー出力データのプラグインへの受け渡しに使用する.

2.4.2　iTrace Atom

iTrace Core に対応する Atom エディタのプラグインである．Github 上で公開されており，ダウンロードして使用できる．閲覧時（2024 年 9 月）は不具合が多く，正常に動作しない状態である． そのため，不具合を修正し，以下の機能を完成させる.

1. 視線座標からソースコード中の行・列への変換機能
2. ソースコード中の行・列からソースコード中の単語への変換機能
3. 注視中の単語をエディタ上でハイライトする機能

2.5　本研究で採用する手法

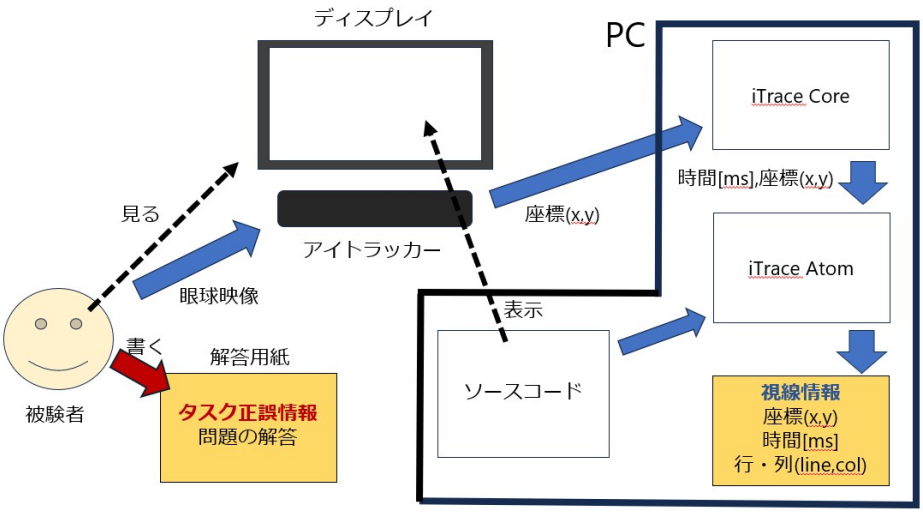

図 2: 実験装置の全体像

図 2 に，本研究で採用する視線データ計測装置の全体図を示す． 吉岡らが提案したマッピング手法を実装し，アイトラッカーで取得した視線座標を構文木中のクラスと結びつけ，被験者一人一人の視線情報を得る．同時に，被験者のタスクへの解答を取得し， タスク正誤情報を得る． その後，クラス毎の注視時間割合を求め，正解・不正解ごとの差異を検証する．

第 3 部　iTrace Atom プラグインの開発

実験で使用する iTrace Atom プラグインは，公開されているソースコードを修正・追記する形で実験に使用可能な機能を開発した． プラグインの処理に関わるソースコードは JavaScript で記述されている．

3.1　視線座標を行・列に変換する関数 getLineCoumn()

完成した関数のソースコードを図??に示す．

この関数は視線座標を point 変数として受け取り，row 変数と column 変数の組を返り値として返す関数である． 上部（2 行目〜16 行目）で定義されている変数を解説する．

3.2　生データ xml から注視時間割合までの処理

実験により得られた生データは xml 形式で保存される．表 3 に，生データが持つパラメータを記す．

表 1: 生データに含まれるデータ一覧

変数名	値の意味	格納される型
bounds	操作中の Atom エディタのウィンドウの縦横サイズ [px] の組	int
editorWorkingArea	操作中のソースコードウィンドウの縦横サイズ [px] の組	int
overscan	操作中のソースコードウィンドウの左上端の座標 [px]	int
currentElement	現在の注視点が位置するソースコード要素	string
offsetX		
columnWidth	ソースコード中の半角文字の横幅 [px]	int
row	出力する列	int
column	出力する行	int

ソースコード 1: getLineColumn 関数のソースコード

```
1         getLineColumn(point) {
2           const { bounds } = this.screen.getPrimaryDisplay();
```

```
3           const editorWorkingArea = this.screen.getPrimaryDisplay().
               workAreaSize;
4           const overscan = {
5             x: bounds.width - editorWorkingArea.width + 8,
6             y: bounds.height - editorWorkingArea.height + 8
7           };
8           const currentElement = this.document.elementFromPoint(
9             point.x - overscan.x,
10            point.y - overscan.y
11          );
12          const offsetX = this.calculateOffsetX();
13          const columnWidth = this.getColumnWidth();
14
15          let row = null;
16          let column = null;
17          if (currentElement === null) {
18            return undefined;
19          }
20          // whitespace
21          if (currentElement.classList.contains('line')) {
22            row = parseInt(currentElement.getAttribute('data-screen-row'),
                   10);
23            column = Math.floor((point.x - offsetX + this.editerElement.
                  getScrollLeft()) / columnWidth);
24          }
25
26          // code element of some sort
27          else if (currentElement.className.includes('syntax') ||
               currentElement.className.includes('leading-whitespace')) {
28            const line = currentElement.closest('.line');
29            row = parseInt(line.getAttribute('data-screen-row'), 10);
30            column = Math.floor((point.x - offsetX + this.editerElement.
                  getScrollLeft()) / columnWidth);
31          }
32
33          // probably looking outside window
34          // todo: map all objects
35          else {
36            return undefined;
37          }
38
39          return { row, column };
40        }
```

第 4 部　実験

本章では，実験の目的，実験に際し行った準備，当日の手順，得られたデータの概要を示す．
実験は 2 章で説明した手法を実装した実験台を 3 台用意し，被験者を 3 人ずつ同時並行で測定する．

4.1　実験の目的

本実験は，以下の目的で行う．

1. Java の基礎知識を有する被験者が，複数のクラスで構成され 100 行以上にわたる比較的長大な Java ソースコードを読解する際の視線情報の収集
2. 上述のソースコード読解時の被験者の Java 理解度の収集

この目的達成のため，以下の手段を用いる．

1. 被験者にソースコードと，それに対応するタスクを提示し，思考・解答中の視線座標をアイトラッカーを用いて収集し，視線情報とする
2. 被験者がタスクに正答した場合はソースコードを理解していると見なし，誤答した場合は理解していないと見なす

4.2　実験対象

実験は，近畿大学工業高等専門学校・総合システム工学科・制御情報コース 4 年生の学生 33 名を対象とする．被験者全員がオブジェクト指向を含む Java プログラミングの授業を受講した経験がある．
なお，被験者全員に以下の事項を伝え，承諾書にサインしていただいた．

1. 実験で使用する装置が身体に害を及ぼさないこと
2. 計測されたデータを個人を特定できないよう十分配慮した上で研究発表に使用すること
3. 被験者となるかは任意であり，実験結果が学校の成績等に一切の影響を与えないこと

4.3　被験者に提示するソースコードとタスク

4.3.1　ソースコード

ソースコードは Java で記述し，Main クラスを含む計 6 つのクラスで構成されるものを準備する． コードは 148 行あり，上部にタスク 3〜5 で使用するクラスを，下部に Main クラスを記述し，行で区切られた範囲に各タスクで使用する文を記述する． （図??）．

図??中の class Object〜からの 7 行が Object クラス，class Animal〜からの 14 行が Animal クラス， class Robot〜からの 26 行が Robot クラス，class Person extends Animal〜からの 15 行が Person クラス， class Cat extends Animal〜からの 14 行が Cat クラスの記述である．Person クラス，Cat クラスは Animal クラスを継承している．

また，図??に示す Main クラスには各タスクで使用するソースコードが区切られて順番に記述されている． 各タスクのソースコードは独立しており，区切りの中のコードと図??のクラス定義文のみを読んでタスクに解答できる．

ソースコード 2: 実験で被験者に提示したソースコード

```
1         class Object {
2             private String name;
3
4             public void setName(String name) {
5                 this.name = name;
6             }
7         }
8
9         class Animal {
10            private String name;
11
12            public void setName(String name) {
13                this.name = name;
14            }
15            public String getName() {
16                return this.name;
17            }
18
19            public void speak() {
20                System.out.println("モー モー！");
21            }
22        }
```

```
23
24          class Robot {
25              private String os;
26              private int x;
27              private int y;
28              private String name;
29
30              public void setName(String name) {
31                  this.name = name;
32              }
33              public String getName() {
34                  return this.name;
35              }
36
37              public void setOs(String os) {
38                  this.os = os;
39              }
40              public void setNumber(int x, int y) {
41                  this.x = x;
42                  this.y = y;
43              }
44
45              public void process() {
46                  int out = x * y;
47                  System.out.println(this.os + "出力：" + out);
48              }
49          }
50
51          class Person extends Animal {
52              private int age;
53
54              public void setAge(int age) {
55                  this.age = age;
56              }
57              public int getAge() {
58                  return age;
59              }
60
61              public void speak() {
62                  int futureAge = age + 5;
63                  System.out.println("人間です！");
64              }
65          }
66
```

```
67          class Cat extends Animal {
68              private int age;
69
70              public int getAge() {
71                  return age;
72              }
73              public void setAge(int age) {
74                  this.age = age;
75              }
76
77              public void speak() {
78                  System.out.println("ニャー！");
79              }
80          }
81
82          public class Main {
83              public static void main(String[] args) {
84          //=====================================================
85                  //１問目（1分）
86                  int x = 3;
87                  if(x > 5){
88                      x = 0;
89                      if(x < 2){
90                          System.out.println("A");
91                      }
92                      else{
93                          System.out.println("B");
94                      }
95                  }
96                  else if(x <= 5){
97                      x = 10;
98                      if(x > 5){
99                          x = 7;
100                         System.out.println("C");
101                     }
102                     else if(x == 7){
103                         System.out.println("D");
104                     }
105                     else{
106                         System.out.println("E");
107                     }
108                 }
109
110         //=====================================================
```

```
111                 //２問目（１分）
112                 int[] indexes = {1, 3, 4, 6};
113                 int[] array = {8, 3, 9, 4, 2, 4, 6, 8, 1, 5};
114 
115                 int sum = 0;
116                 for(int i = 0; i < indexes.length; i++){
117                     sum += array[indexes[i]];
118                 }
119 
120                 System.out.println(sum);
121 
122         //=======================================================
123                 //３問目(2分)
124                 Robot robot = new Robot();
125                 robot.setOs("Windows");
126                 robot.setNumber(2, 3);
127                 robot.process();
128 
129         //=======================================================
130                 //４問目(2分)
131                 Person person = new Person();
132 
133                 person.setName("Taro");
134                 person.setAge(25);
135 
136         //=======================================================
137                 //５問目(2分)
138                 Animal animal1 = new Person();
139                 Animal animal2 = new Cat();
140                 Animal animal3 = new Animal();
141 
142                 animal1.speak();
143                 animal2.speak();
144                 animal3.speak();
145 
146         //=======================================================
147             }
148         }
```

4.3.2 タスク

タスクは全５問用意する．うち２問がオブジェクト指向の概念を伴わない Main クラス内で完結するタスク，うち ３ 問がオブジェクト指向の概念を伴うタスクである． ソース

コード 3〜ソースコード 7 にタスクの詳細を示す．

タスク 1:ソースコード 3 に示すコードの読解を課す．変数代入と if 文による条件分岐を組み合わせたタスク．標準出力の出力結果を答えさせる．オブジェクト指向の概念を伴わない．

ソースコード 3: タスク 1 のソースコード

```
1         //1 問目（1分）
2         int x = 3;
3         if(x > 5){
4             x = 0;
5             if(x < 2){
6                 System.out.println("A");
7             }
8             else{
9                 System.out.println("B");
10            }
11        }
12        else if(x <= 5){
13            x = 10;
14            if(x > 5){
15                x = 7;
16                System.out.println("C");
17            }
18            else if(x == 7){
19                System.out.println("D");
20            }
21            else{
22                System.out.println("E");
23            }
24        }
```

タスク 2:ソースコード 4 に示すコードの読解を課す．配列の参照を使用したタスク．標準出力の出力結果を答えさせる．オブジェクト指向の概念を伴わない．

ソースコード 4: タスク 2 のソースコード

```
1         // 2 問目（1分）
2         int[] indexes = {1, 3, 4, 6};
3         int[] array = {8, 3, 9, 4, 2, 4, 6, 8, 1, 5};
4
5         int sum = 0;
6         for(int i = 0; i < indexes.length; i++){
7             sum += array[indexes[i]];
8         }
9
10        System.out.println(sum);
```

タスク 3:ソースコード 5 に示すコードの読解を課す．1 種類のクラスを使用し，オブジェクト生成と外部からのメソッド実行を含むタスク．標準出力の出力結果を答えさせる．オブジェクト指向の概念を伴う．

ソースコード 5: タスク 3 のソースコード

```
1         // 3 問目(2分)
2         Robot robot = new Robot();
3         robot.setOs("Windows");
4         robot.setNumber(2, 3);
5         robot.process();
```

タスク 4:ソースコード 6 に示すコードの読解を課す．2 種類のクラスを使用し，オブジェクト生成とセッターの呼び出しを含むタスク．2 種類のクラスは片方がもう片方を継承する関係にある． ソースコード内で，それぞれのオブジェクトのセッターが定義された行を答えさせる．オブジェクト指向の概念を伴う．

ソースコード 6: タスク 4 のソースコード

```
1         // 4 問目(2分)
2         Person person = new Person();
3
4         person.setName("Taro");
5         person.setAge(25);
```

タスク 5:ソースコード 7 に示すコードの読解を課す．3 種類のクラスを使用し，メソッドのオーバーライドを含む．標準出力の出力結果を答えさせる．オブジェクト指向の概念を伴う．

ソースコード 7: タスク 5 のソースコード

```
1         // 5 問目(2分)
2         Animal animal1 = new Person();
3         Animal animal2 = new Cat();
4         Animal animal3 = new Animal();
5
6         animal1.speak();
7         animal2.speak();
8         animal3.speak();
```

4.4 実験台の構成

1つの実験台につき一人を座らせ，視線運動の測定を行う． 本実験では，測定の効率化のため被験者 3 人ずつ同時に測定を行うため，実験台は計 3 台用意する． 各実験台を構成する機器を以下に示す．

1. ディスプレイ
   3 台の実験台全てに，画面サイズ 1920×1200 ピクセルのディスプレイを使用する．
2. コンピュータ
   3 台の実験台全てに，Windows11 がセットアップされたコンピュータを使用する．コンピュータにはあらかじめ Tobii Pro Eye Tracker Manager，iTrace Core と，iTrace Atom プラグインがインストールされた Atom エディタをインストールする．
3. アイトラッカー
   3 台の実験台のうち，2 台に Tobii Pro Spark を搭載し，1 台に Tobii Pro Nano を搭載する． 3 台全てにおいてディスプレイ画面下に設置する．
4. マウス
   3 台の実験台全てに，被験者にエディタを操作していただくため用意する．
5. 顎台
   3 台の実験台全てに，被験者の顔を固定可能な顎台を用意する． 本実験は測定時間が約 8 分と長く，またタスクの間には解答用紙記述のため画面から目を離す必要があるため， 被験者が顔の適切な位置を見失うことが想定される．そのため，顔の位置を固定可能な台を用意し， アイトラッキングの精度を保つことが必要となる．

図 3 に，実験部屋に設置した 3 台の実験台の様子を示す． ディスプレイの右上端に A,B,C の記号が書かれた紙を貼り，各実験台を区別する． 被験者は椅子に座り，顎台に顔を置いた状態で実験を受ける． 机上のキーボードは実験運営側の操作に使用する．

図 3: 実験部屋の様子

4.5　実験手順

実験は，以下の手順で行う．

1. 実験台のセットアップ
2. 承諾書記入
3. キャリブレーション
4. 実験の詳細説明
5. 練習問題
6. 5 つのタスク提示・視線座標計測

実験手順を詳細に解説する．

4.5.1　実験台のセットアップ

被験者の交代の度に実験台を図 4 のようにセットアップする． 画面には Atom エディタ，iTrace Core，Tobii Pro Eye Tracker Manager のウィンドウを表示させ， 後の測定が円滑にする．iTrace Core は Start/Stop ボタンが常に見える位置に配置する． Atom エディタには練習問題のソースコードを表示しておく． 机上には解答用紙と承諾書を配置しておく．

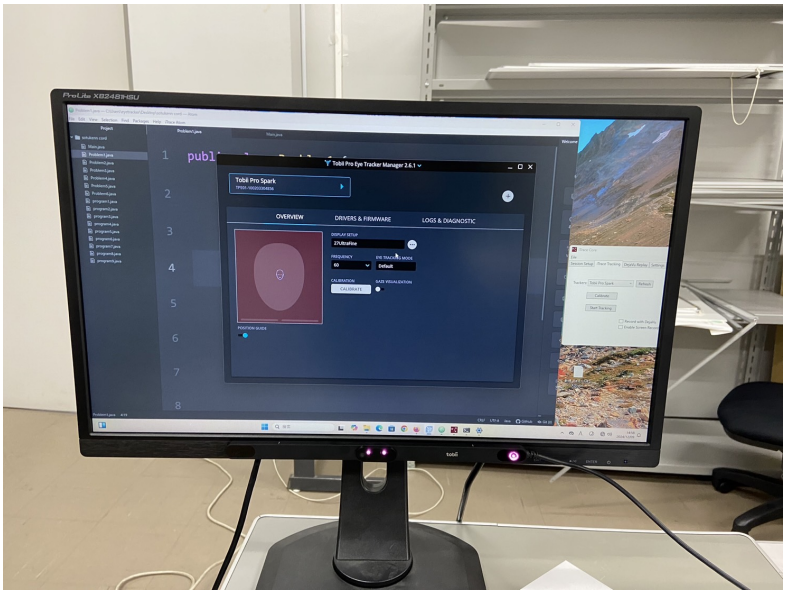

図 4: セットアップされた実験台の画面

4.5.2　承諾書記入

被験者に承諾書の内容を確認していただき，サインしていただく． 承諾書はセットアップ時に机上に配置し，サインが終わり次第回収する．

4.5.3　キャリブレーション

まず，Tobii Pro Eye Tracker Manager が示す適切な顔の位置に顎台を固定する． 固定ができ次第キャリブレーションを行う． 要再測定の表示が出た場合は再測定をする．

4.5.4　実験の詳細説明

被験者に以下の事項をお伝えする．

1. ソースコードを読解しながらタスクを解くこと
2. 測定終了まで顎台から顔を離さず，首を動かさないこと
3. ソースコード読解時はエディタのスクロール操作のみ行い，書き込みはしないこと
4. 問題が解き終わり次第，挙手で合図を出して画面から目をそらし，解答用紙に記入すること

4.5.5　練習問題

計測の前に，被験者に実験に慣れていただくことと，データが正しく取れることの確認のため，練習問題を実施する． 問題の内容は練習問題用ソースコードの全ての行を読むこととする． 練習問題終了時にデータ保存先フォルダを確認し，正しく保存されているかを確認する．

4.5.6　5 つのタスク提示・視線座標計測

測定用ソースコードとタスクを提示し，被験者に解いていただく． 各タスクごとに測定用ソースコードの Main クラス内に記述された読解開始の合図と同時にタスクの内容を口頭で説明する．

4.6 実験結果

実験により，計 33 人の視線座標データを取得した．そのうち，正確に保存されなかったデータと測定時のプログラムに誤りがあったデータを除き， 後の分析に使用できるデータを抽出する．抽出後のタスクごとのデータ数と，正誤数を表 2 に示す． なお，タスク 5 において，3 つある解答のうち，それぞれの内容はあっているものの記述順が異なっている場合に「惜しい」判定とする．

表 2: 実験により取得された使用可能データと，各タスクごとの正誤数

タスク名	使用可能なデータ数	正答数	誤答数	惜しい
タスク 1	26	6	20	-
タスク 2	10	3	7	-
タスク 3	10	3	7	-
タスク 4-1	26	2	24	-
タスク 4-2	26	7	19	-
タスク 5	26	13	8	5

第 5 部　データの分析

本章では，実験で得られたデータを加工し，6 つのクラスごとの注視時間割合に変換する過程と，その後の分析の一連の流れを示す． データは加工の過程で生データ xml，変数付き csv，注視時間割合の順に変換される． 変換後の 6 次元の注視時間割合ベクトルを主成分分析で 3 次元，2 次元に圧縮し，3D プロット，2D プロットに描画して観察する．

5.1　生データ xml から注視時間割合までの処理

実験により得られた生データは xml 形式で保存される．表 3 に，生データが持つパラメータを記す．

表 3: 生データに含まれるデータ一覧

パラメータ名	値の意味	値の例
screen_width	ディスプレイ画面の横 px 数	1920
screen_height	ディスプレイ画面の縦 px 数	1200
plugin_type	プラグインの種類	ATOM
gaze	視線情報	後述

gaze パラメータはアイトラッカーのサンプリング数だけ存在し，それぞれ表 4 のパラメータを持つ． なお，アイトラッカーのサンプリングレートは Tobii Pro Spark，Tobii Pro Nano ともに 60Hz である．

表 4: 生データに含まれる視線情報

パラメータ名	値の意味	値の例
event_id	2	133784518694431017
plugin_time	サンプリングの UNIX 時間 [ms]	1733978269442
x	ディスプレイ画面上の x 座標 [px]	480
y	ディスプレイ画面上の y 座標 [px]	418
source_file_line	ソースコード中の行	2
source_file_col	ソースコード中の列	9
word	視線が位置していた単語	int
gaze_target	視線が位置していたファイルの名前	Problem1.java
gaze_target_type	対象ファイルの拡張子	java
source_file_path	対象ファイルのファイルパス	Problem1.java
editor_line_height	エディタの行の高さの設定値	40
editor_font_height	エディタのフォントサイズの設定値	120

上述の生データから特定の情報を抜き出し，プログラムで扱いやすいよう csv に加工する． この際，サンプリング間の速度と AOI(Area Of Instance) も計算し，csv の列に加える．計算後の変数付き csv の主なパラメータを表 5 に示す．

表 5: 変数付き csv の主なパラメータ

パラメータ名	値の意味	値の例
time	計測開始からの経過時間 [ms]	123
x	ディスプレイ画面上の x 座標 [px]	480
y	ディスプレイ画面上の y 座標 [px]	418
line	ソースコード中の行	2
col	ソースコード中の列	9
velocity	速さ [px/ms]	0.5893286
AOI	クラス番号	1

求めた変数付き csv から，6 つのクラスごとの注視時間割合ベクトルを算出する．

5.2　3D・2D プロットの観察

　求めた被験者ごとの注視時間割合ベクトルを主成分分析にて 3 次元，2 次元に落とし，それぞれ 3D プロット,2D プロットで可視化する． 各被験者をドットとし，タスクの正誤ごとに色分けして表示する．色分けは，正解者が青，不正解者が赤，惜しいが緑とする．

5.2.1　タスク 3

タスク 3 の分布について，図 5 に 3D プロットを，図 6 に 2D プロットを示す．図 5 の 3D プロットについて，正解者は一定の場所に固まり，不正解者は表示範囲全体に広がっている．また，図 6 の 2D プロットについて，正解者は右上に固まっており，不正解者は表示範囲全体に広がっている．

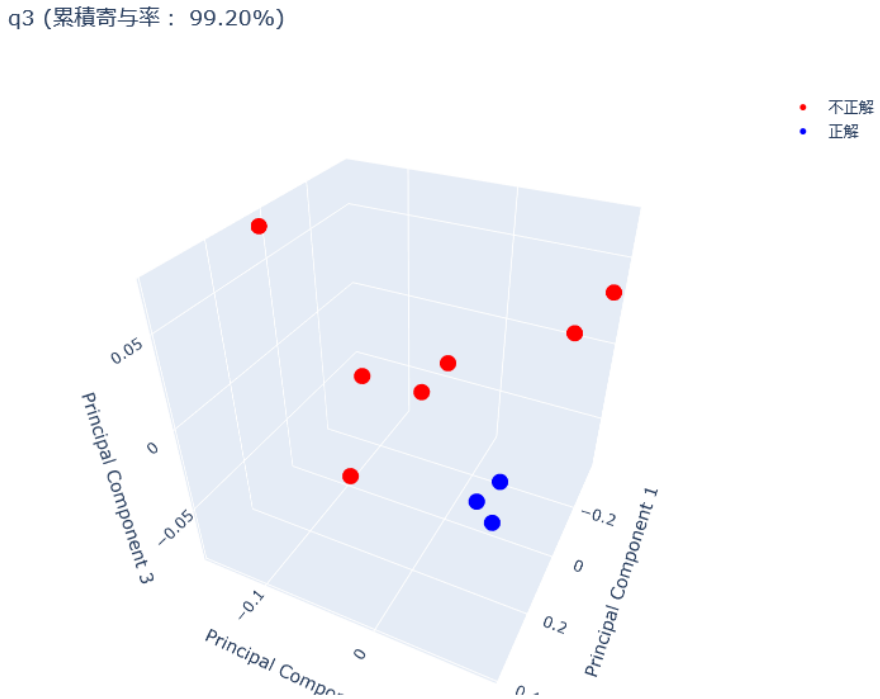

図 5: タスク 3 の分布（3D プロット）

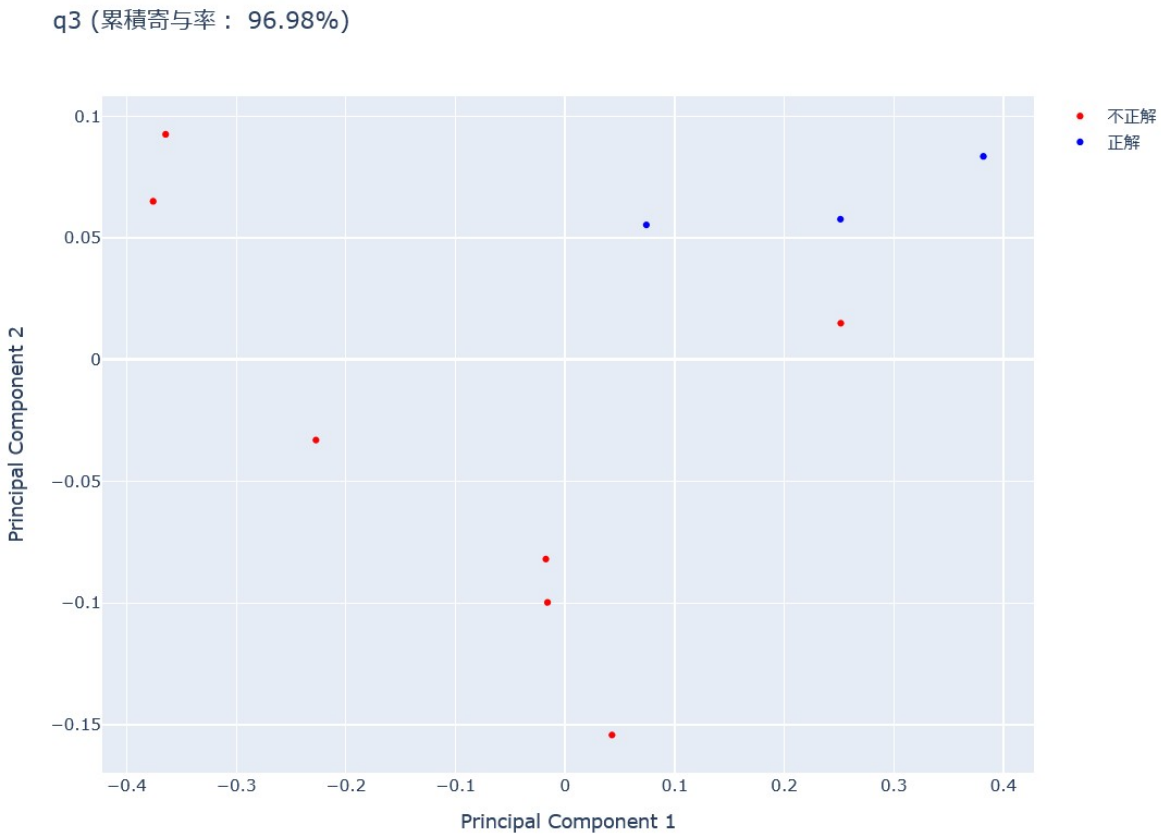

図 6: タスク 3 の分布（2D プロット）

5.2.2　タスク 4

タスク 4 の分布について，図 7 に 3D プロットを，図 8 に 2D プロットを示す．図 7 の 3D プロットについて，正解者は中央右寄りに分布し，不正解者は中央から右側にかけて分布している． また，図 8 の 2D プロットについて，正解者は中央左寄りに固まっており，不正解者は中央から左側にかけて広がっている．

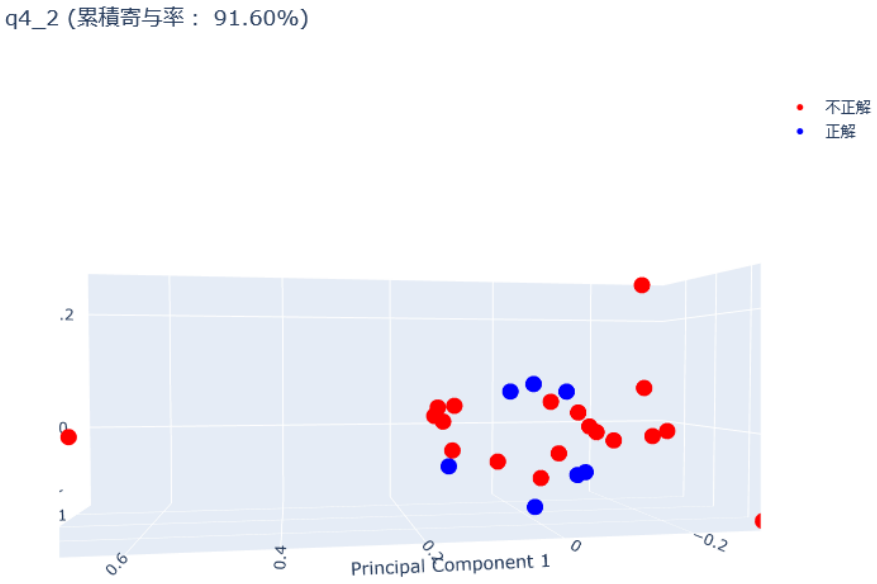

図 7: タスク 4-2 の分布（3D プロット）

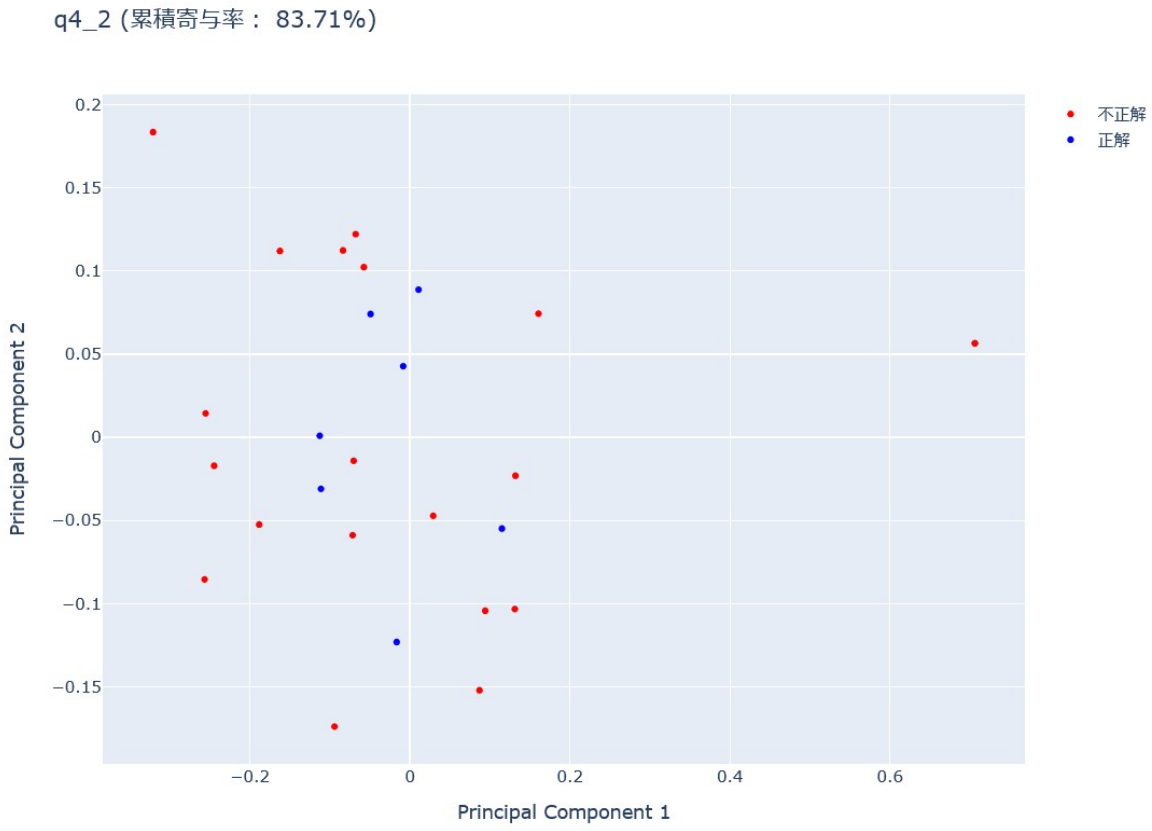

図 8: タスク 4-2 の分布（2D プロット）

5.2.3　タスク 5

タスク 5 の分布について，図 9 に 3D プロットを，図 10 に 2D プロットを示す． 図 9 の 3D プロットについて，正解者は上部に広く分布し，不正解者と惜しいは中央と右側に分かれて分布している． また，図 10 の 2D プロットについて，正解者は中央から左側にかけて分布し，不正解者は表示範囲に散らばり，惜しいは中央右寄りと左側に分布している．

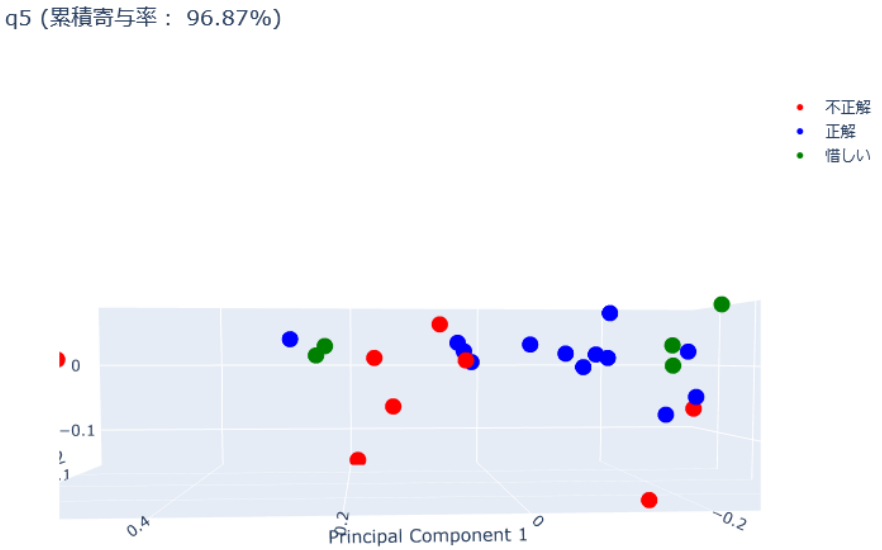

図 9: タスク 5 の分布（3D プロット）

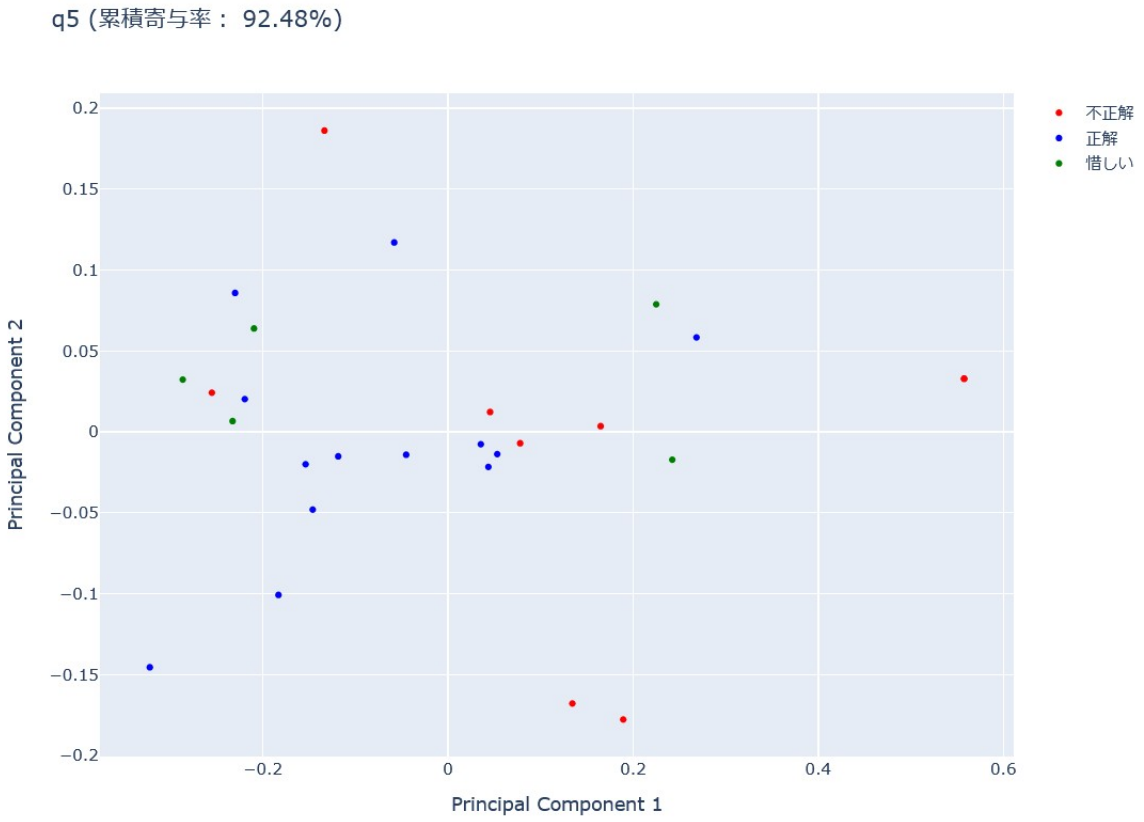

図 10: タスク 5 の分布（2D プロット）

5.2.4 考察

タスク 3，タスク 4，タスク 5 において，正答者のプロットは不正解者のものと比較して固まって位置する傾向にあることが分かる． 特に，タスク 3 においては正解者と不正解者の分布範囲に明らかな違いが見られる．これは，タスク 3 では Robot クラスのみが使用されるために， ソースコードの動作を理解している人は Main クラスと Robot クラスの範囲に集中していることが原因と考えられる． 逆に，タスク 5 においては正解者の分布が他のタスクと比較して散らばっている．これはタスク 5 が 3 つのクラスの読解を必要とするため， ソースコードの動作を理解している被験者においても視線がソースコードのクラス定義文の範囲全体に散らばりやすく，理解していない被験者との差が生まれにくいことが原因と考えられる． また，全てのタスクにおいて不正解者の分布が全体に散らばっている傾向にあるが，これは不正解者は共通した特定の読解方法を持たず，おのおのが異なる読解を行うことが原因と考えられる．

第 6 部　結論

本研究により，オブジェクト指向を取り入れたタスクにおいて，正答者の視線運動は 3 次元グラフ上のある一定の範囲に固まる傾向があり， 不正答者のものはよりばらつきが大きいことが分かった．主成分分析での次元圧縮の際に算出した累積寄与率は 90%以上と高く， 圧縮前のパラメータでもおおむね同様の傾向であるといえる．このことから，オブジェクト指向を取り入れたソースコードを用いたタスクにおいてもプログラミング理解度に応じて視線運動に差異が生じていることが分かった．

次に，分布の違いを利用して未知データの分類が可能といる意見について考察する．今回用いたクラスごとの注視時間割合ベクトルのデータでは，上述の通り正解者と不正解者で範囲がきれいに分かれているわけではなく，正解者が一定の範囲に集まり，不正解者がより広い範囲に分散するという結果となった． そのため，単純な K-means 法や K-medois 法などのクラスタリング手法では分類が困難であるといえる．

今後は，より多くのデータを集めて正答者の視線運動の傾向をつかみ， その傾向から外れている場合に読解者がコードを理解していない可能性を検知できないか検証することが求められる．

謝辞

本研究は，研究生活を通して熱心にご指導賜りました，指導教員の大阪公立大学大学院情報学研究科・基幹情報学専攻の大野修一教授，共同で研究に励むと同時に，研究を導いて下さりました岩佐英彦先生，修士論文審査において貴重なご助言を賜りました大阪公立大学大学院情報学研究科・基幹情報学専攻の戸出英樹教授，上野敦志講師， 視線測定実験の共同運営にご尽力頂きました近畿大学工業高等専門学校・総合システム工学科・制御情報コース 5 年生の岡村晏志様， および視線測定実験にご同意・ご協力頂きました近畿大学工業高等専門学校・総合システム工学科・制御情報コース 4 年生 33 名の協力により得られた成果です. ここに記して謝意を表します.

参考文献

[1] 文部科学省,” 小学校プログラミング教育の手引（第三版）”, 文部科学省,https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1375607.htm,（参照：2025-2-13）

[2] 大学入試センター,” 情報 サンプル問題”, 文部科学省,https://www.mext.go.jp/content/20211014-mxt_daigakuc02-000018441_9.pdf,（参照：2025-2-13）

[3] 泰地 酒井, 昌一 浦野, 視線分析の傾向分析による特徴抽出, 人工知能学会全国大会論文集,2021

[4] 亮 花房, 慎平 松本, 雄介 林, 宗 平嶋, 視線運動を用いたプログラム読解パターンのデータ依存関係に基づく分析ーー代入演算と算術演算で構成されるプログラムを対象としてーー, 教育システム情報学会誌,2018

[5] 吉岡春彦, 上野秀剛, 構文木と視線移動の自動マッピング手法を用いたプログラム理解過程の分析, ソフトウェアエンジニアリングシンポジウム 2023,2023

[6] Tobii Pro Spark https://www.tobii.com/ja/products/eye-trackers/screen-based/tobii-pro-spark

[7] Tobii Pro Nano https://www.tobii.com/ja/products/discontinued/tobii-pro-nano

[8] tobii,” アイトラッキングの基礎 1　アイトラッカーの仕組み”,tobii 公式サイト,https://www.tobii.com/ja/blog/how-eye-tracking-working,（参照：2025-2-21）

[9] Tobii Pro Eye Tracker Manager https://connect.tobii.com/s/etm-download

[10] iTrace https://www.i-trace.org/

[11] ATOM https://atom-editor.cc/