

スマートフォンのモーションセンサを利用した 個人認証アプリケーションの開発

情 11-170 高坂 賢佑

目次

第 1 章	使用するセンサについて	7
1.1	加速度センサ	7
1.2	ジャイロセンサ	7
第 2 章	先行研究	9
2.1	坂本の研究	9
2.2	兎澤の研究	9
第 3 章	本研究のシステム	10
3.1	本研究の概要	10
3.2	システムの概要	10
3.2.1	新規登録モード	12
3.2.2	認証試験モード	14
3.2.3	データ閲覧モード	16
3.3	先行研究からの改善点	18
3.3.1	モーションデータの増幅機能	18
3.3.2	フーリエ変換を用いたローパスフィルタ	19
3.3.3	モーション取得時のズレを修正する機能	22
3.3.4	モーション取得時のインターバル及びヴァイブレーション機能	23
第 4 章	実験と考察	24
4.1	実験方法	24
4.2	実験結果	24
4.3	考察	25
4.4	課題	25
第 5 章	おわりに	26

付 録 A	プログラムより抜粋したもの	29
A.1	ズレ修正の判断部分	29
A.2	ズレ修正アルゴリズム	31
付 録 B	実験結果詳細	34
付 録 C	プログラム本体	36
C.1	src/com/example/motionauth/Start.java	36
C.2	src/com/example/motionauth/Registration/RegistNameInput.java	42
C.3	src/com/example/motionauth/Registration/RegistMotion.java	45
C.4	src/com/example/motionauth/Authentication/AuthNameInput.java	64
C.5	src/com/example/motionauth/Authentication/AuthMotion.java	68
C.6	src/com/example/motionauth/ViewDataList/RegistrantList.java	79
C.7	src/com/example/motionauth/ViewDataList/ViewRegisteredData.java	82
C.8	src/com/example/motionauth/ViewDataList/ViewRegisteredRData.java	87
C.9	src/com/example/motionauth/ViewDataList/ViewAuthRData.java	90
C.10	src/com/example/motionauth/Processing/Formatter.java	93
C.11	src/com/example/motionauth/Processing/Amplifier.java	95
C.12	src/com/example/motionauth/Processing/Calc.java	98
C.13	src/com/example/motionauth/Processing/Correlation.java	101
C.14	src/com/example/motionauth/Processing/CorrectDeviation.java	112
C.15	src/com/example/motionauth/Processing/CipherCrypt.java	118
C.16	src/com/example/motionauth/Lowpass/Fourier.java	125
C.17	src/com/example/motionauth/Utility/ConvertArrayAndString.java	130
C.18	src/com/example/motionauth/Utility/Enum.java	132
C.19	src/com/example/motionauth/Utility/LogUtil.java	132
C.20	src/com/example/motionauth/Utility/ManageData.java	135
C.21	res/layout/activity_start.xml	152
C.22	res/layout/activity_regist_name_input.xml	153
C.23	res/layout/activity_regist_motion.xml	154
C.24	res/layout/activity_auth_name_input.xml	157
C.25	res/layout/activity_auth_motion.xml	158
C.26	res/layout/activity_registrant_list.xml	161

C.27 res/layout/activity_view_registered_data.xml	162
C.28 res/layout/activity_view_registered_rdata.xml	162
C.29 res/layout/activity_view_auth_rdata.xml	163
C.30 res/layout/seekdialog.xml	163
C.31 res/menu/regist_motion.xml	164
C.32 res/values/configs.xml	165
C.33 res/values/strings.xml	165
C.34 res/values/styles.xml	166
C.35 AndroidManifest.xml	166

目 次

1.1	モーションセンサの座標系イメージ	8
3.1	システム動作フロー図	11
3.2	スタート画面	11
3.3	モード選択ダイアログ	11
3.4	ユーザ名入力画面	12
3.5	エラー通知	12
3.6	モーション登録画面	12
3.7	処理待ちダイアログ	14
3.8	登録完了ダイアログ	14
3.9	登録失敗ダイアログ	14
3.10	ユーザ名入力画面	15
3.11	認証試験画面	15
3.12	エラー通知	15
3.13	認証成功ダイアログ	17
3.14	認証失敗ダイアログ	17
3.15	ユーザ名一覧画面	17
3.16	モーションデータ一覧画面	17
3.17	新規登録モードメニュー画面	20
3.18	増幅器設定ダイアログ	20
3.19	増幅前のデータ	20
3.20	増幅後のデータ	20
3.21	ローパスフィルタ前のデータ	22
3.22	ローパスフィルタ後のデータ	22
3.23	ズレ修正前のデータ	23

3.24 ズレ修正後のデータ	23
--------------------------	----

はじめに

スマートフォンが徐々に普及しつつある現在，スマートフォンの個人認証方法は画面上に表示されるソフトウェアキーボードのテンキーを用いたパスコード認証が大部分を占めている．しかし，この認証方法は画面ロックを解除するたびに画面に表示されたソフトウェアキーボードを目で見て指でタッチして操作する必要があるため，ユーザにとって煩雑な作業である．また，あらかじめ決められた文字種の中から一つずつ選択したものを元にパスコードを構築していくという性質上，パターン数が限られ自由度が限定されてしまう．

そこで，本研究ではパスコード認証が抱える認証の煩雑さを解消し，かつ自由度が高くより直感的に個人認証を行えるアプリケーションを開発する．このアプリケーションには，一般的なスマートフォンに搭載されている加速度センサとジャイロセンサを用いる．

第1章 使用するセンサについて

本アプリケーションには2種類のモーションセンサを利用する。

1.1 加速度センサ

加速度センサとは、X軸、Y軸、Z軸の基準軸に対して直線運動の加速度をそれぞれ検出し、値として取り出すことのできるセンサである。ここでいう加速度とは端末における単位時間あたりの速度の変化率のことを指し、図 1.1 における直線で示した矢印の方向に端末を動かした場合が正の値、逆が負の値をとる。

1.2 ジャイロセンサ

ジャイロセンサとは、X軸、Y軸、Z軸の基準軸に対して回転運動の角速度をそれぞれ検出し、値として取り出すことのできるセンサである。ここでいう角速度とは端末における単位時間あたりの回転角のことを指し、図 1.1 における橙色で示した回転の方向に端末を動かした場合が正の値、逆が負の値をとる。

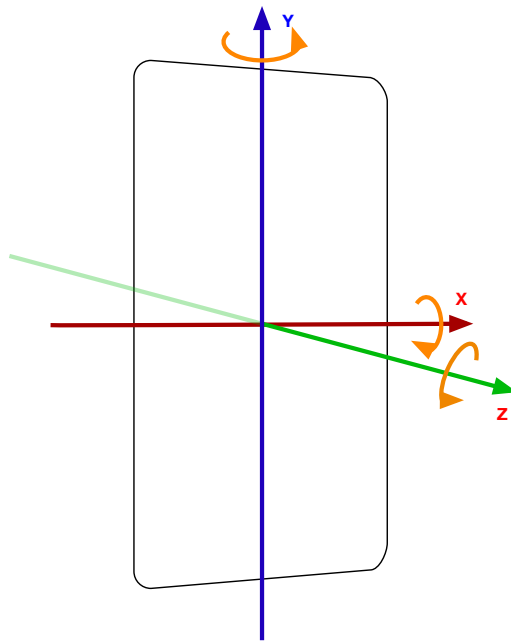


図 1.1: モーションセンサの座標系イメージ

第2章 先行研究

2.1 坂本の研究

坂本の研究[1]では、ユーザが入力したモーションの数値化に加速度センサを用い、あらかじめ保存しておいた複数のジェスチャパターンのデータと認証時に入力したデータをパターンマッチング方式のアルゴリズムを用いて比較することで個人認証を行った。

しかし、このプログラムは扱うジェスチャによって認証率が高いものと低いものに二分化する傾向が見られるという問題点があった。

2.2 兎澤の研究

兎澤の研究[2]では、モーションの数値化に加速度センサとジャイロセンサを用い、認証システムの中核に相関関数を用いたシステムを開発し、ユーザに3回入力させたモーションの平均値データと認証時に入力したデータの類似性を調べることで個人認証を行った。これにより、坂本の研究で指摘されていた成功率の二分化や立体的な動きへの対応を可能にし、モーションの対応幅を広げることができた。

しかし、全体的な認証成功率が低く、特に手首のスナップを用いるような動きの小さいモーションに対して認証率が特に低く出るなど、対応できるモーションに限りがあるという問題点が指摘されていた。

第3章 本研究のシステム

3.1 本研究の概要

本研究では兎澤の研究で挙げられていた，全体的な認証成功率の低さや対応できるモーションに限りがあるという点を改善することを目標とする．具体的には，より幅広いモーション，特に手首のスナップを用いるような比較的動きの小さいモーションに対しての個人認証の全体的な認証成功率の向上を目指し，実用レベルに近いアプリケーションの開発を行う．

3.2 システムの概要

本研究では，先行研究をもとに Android デバイス上で動作するアプリケーションとしてシステムを構築した．システムの動作フローを図 3.1 に示す．

アプリケーション起動時は，図 3.2 のような起動画面が表示される．ここで Start ボタンを押すことで，図 3.3 のようなモード選択ダイアログが表示される．

ユーザはまず，新規登録モードにおいて個人認証に用いる鍵情報となるモーションをユーザ名と共に登録する．このモードでは，ユーザに登録したい同一のモーションを 3 回入力してもらう．入力された 3 回のモーションが同一のモーションであると確認できた場合に，この平均値をユーザのモーションデータとして登録する．

認証試験モードでは，事前に新規登録モードにおいてモーションデータを登録したユーザ名を入力し，該当ユーザが登録されていると確認できた場合にのみユーザにモーションを 1 回入力してもらう．入力されたモーションデータと指定したユーザ名で登録されたモーションデータとの相関を取ることによって個人認証を行う．

データ閲覧モードでは，新規登録モードにおいて登録したユーザ名およびモーションデータをリスト形式で閲覧することが出来る．

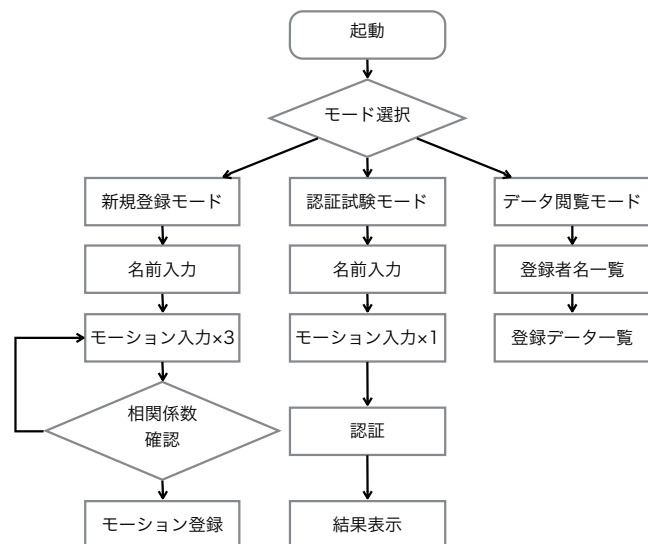


図 3.1: システム動作フロー図

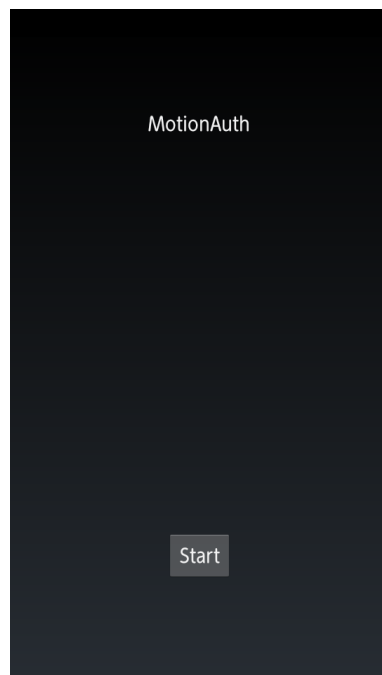


図 3.2: スタート画面

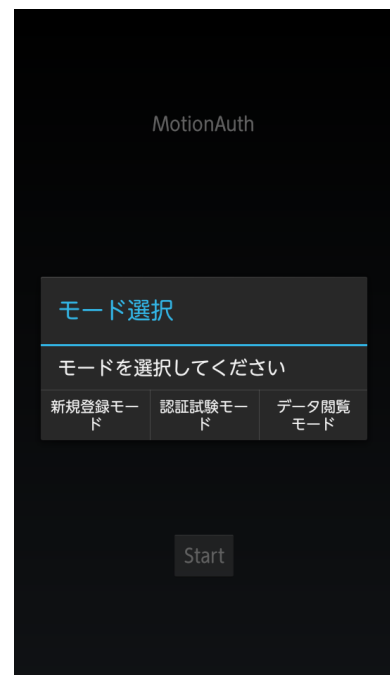


図 3.3: モード選択ダイアログ

3.2.1 新規登録モード

新規登録モードでは、まず図 3.4 の画面において登録したいユーザの名前を入力する。この画面において OK ボタンを押した際にテキストフィールドが空であれば、図 3.5 のような通知を表示してユーザに名前の再入力を促す。テキストフィールドが空でなければ、次の図 3.6 の画面においてモーションの登録を行う。OK ボタンを押した際の名前の入力値チェックを行うコードをソースコード 3.1 に示す。

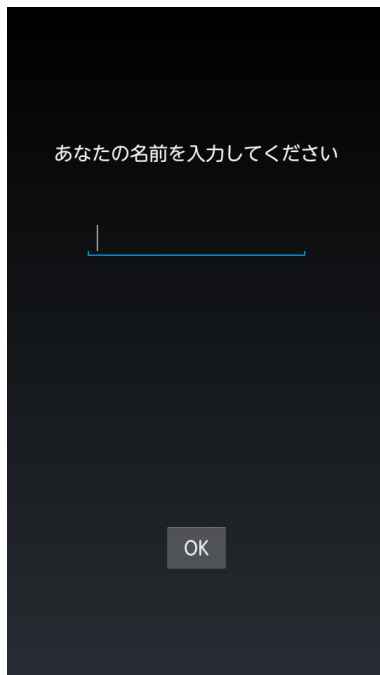


図 3.4: ユーザ名入力画面

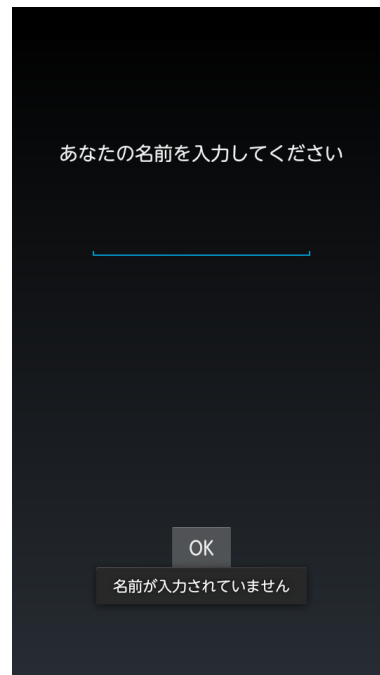


図 3.5: エラー通知

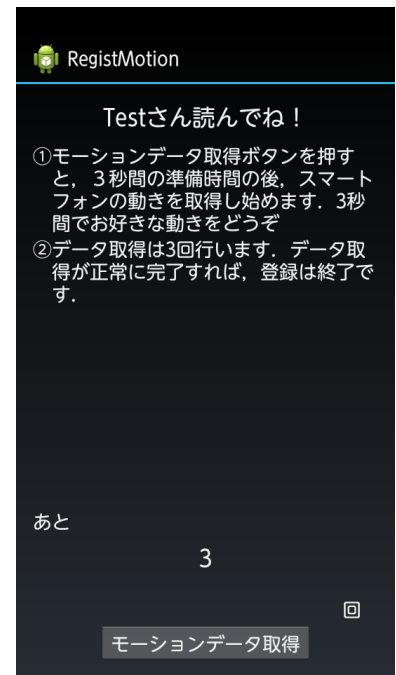


図 3.6: モーション登録画面

ソースコード 3.1: 入力値チェック

```

1 public class RegistNameInput extends Activity {
2     // ユーザが入力した文字列（名前）を格納する
3     public static String name;
4     ...
5     private void nameInput () {
6         final EditText nameInput = (EditText) findViewById (R.id.
            nameInputEditText);
7
8         nameInput.addTextChangedListener(new TextWatcher() {
9             ...
10            public void afterTextChanged (Editable s) {
11                // ユーザの入力した名前を name に格納
12                if (nameInput.getText() != null) name = nameInput.getText().
                    toString().trim();
13            }
14        });
15    }
16 }

```

```
14     });
15     ...
16     // OKボタンを押した時に、次のアクティビティに移動
17     final Button ok = (Button) findViewById(R.id.okButton);
18
19     ok.setOnClickListener(new View.OnClickListener() {
20         @Override
21         public void onClick (View v) {
22             // nameが入力されているかの確認
23             if (name.length() == 0) {
24                 Toast.makeText(RegistNameInput.this, "名前が入力されていませ
                ん", Toast.LENGTH_LONG).show();
25             }
26             else {
27                 RegistNameInput.this.moveActivity("com.example.motionauth",
                "com.example.motionauth.Registration.RegistMotion",
                true);
28             }
29         }
30     });
31 }
32 }
```

モーション登録画面においてモーションデータ取得ボタンを押すと、3秒間のインターバルを挟んだ後にモーションデータの取得を3秒間行う。この際、画面の下部にそれぞれの経過秒数を表示し、ヴァイブレーションにて1秒毎の時間経過を知らせるようにしている。

モーションデータの取得は加速度データ、ジャイロデータそれぞれを0.03秒ごとにセンサから取得し、 $X \cdot Y \cdot Z$ 軸のそれぞれから100個ずつ、計600個を1回分として取得する。

モーションデータの取得が3回行われると図3.7のような計算処理待ちダイアログが表示され、データの加工が行われる。あらかじめ増幅器の閾値を設定しておき、取得したデータの振れ幅が閾値より1回でも小さい場合はモーションの動きが小さいと判断し、全てのデータに振れ幅の増幅処理を行う。次にフーリエ変換を用いたローパスフィルタ処理によって、モーション取得時の手の細かなブレなどから生じるデータに対する影響を取り除く。ローパスフィルタ処理が終われば、取得したデータが同一のものであるかの確認を行う。同一のものであると確認されなければ、モーションデータの取り直しを行う。同一のものであると確認されれば、モーション取得時に生じる時間的なズレを必要に応じて修正する。ズレ修正の処理が終われば3回分のデータ間の相関係数を算出し、相関が認められた場合は図3.8のような登録完了ダイアログを表示する。このダイアログのOKボタンを押すことで、取得した3回分のデータの平均値データと増幅量を保存し、起動画面に移動する。相関が認められなかった場合

は、図 3.9 のような登録失敗ダイアログを表示する。このダイアログの OK ボタンを押すことでプログラム内部のモーションデータ取得カウンタが初期化され、モーションデータの取り直しをさせる。

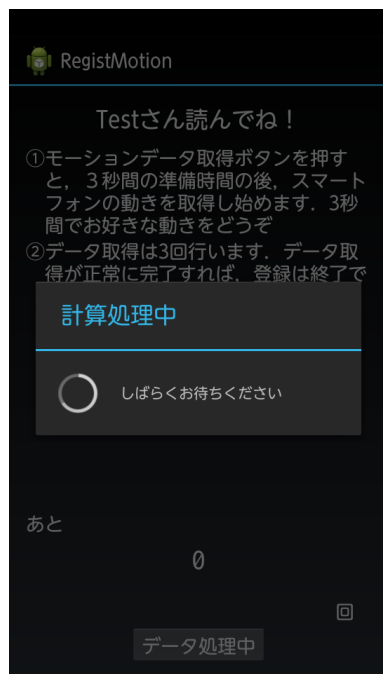


図 3.7: 処理待ちダイアログ

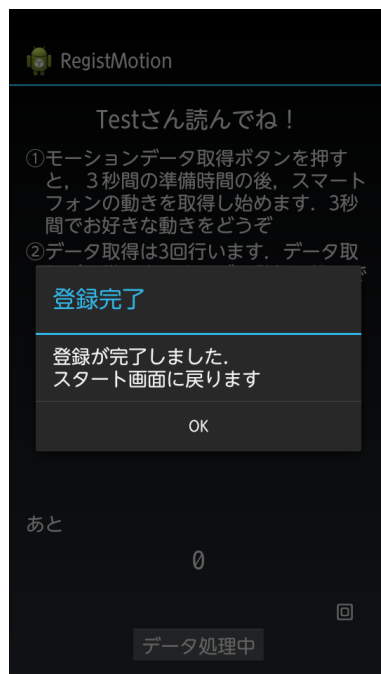


図 3.8: 登録完了ダイアログ

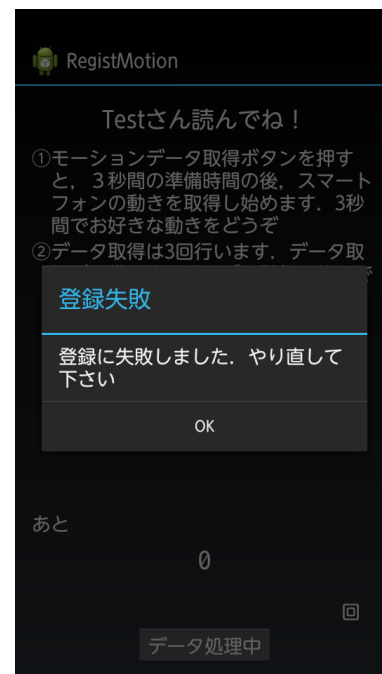


図 3.9: 登録失敗ダイアログ

3.2.2 認証試験モード

認証試験モードでは、まず図 3.10 の画面において新規登録モードであらかじめ登録されたユーザ名を入力する。指定されたユーザ名で既にモーションの登録がなされていることが確認できた場合にのみ、図 3.11 の画面で個人認証を行う。モーションの登録がなされていない場合、図 3.12 のようなダイアログを表示する。指定されたユーザ名で既にモーションの登録がなされているかを確認するコードをソースコード 3.2 に示す。

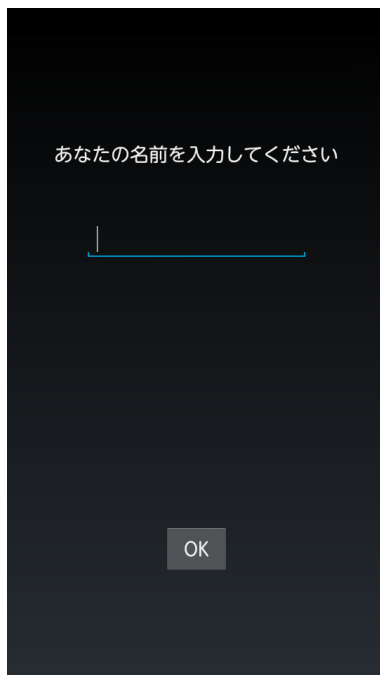


図 3.10: ユーザ名入力画面

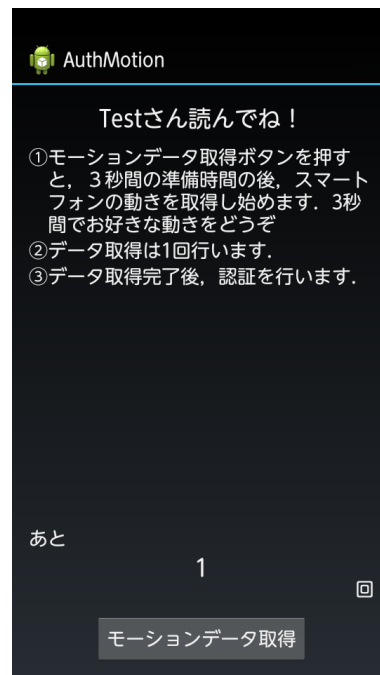


図 3.11: 認証試験画面

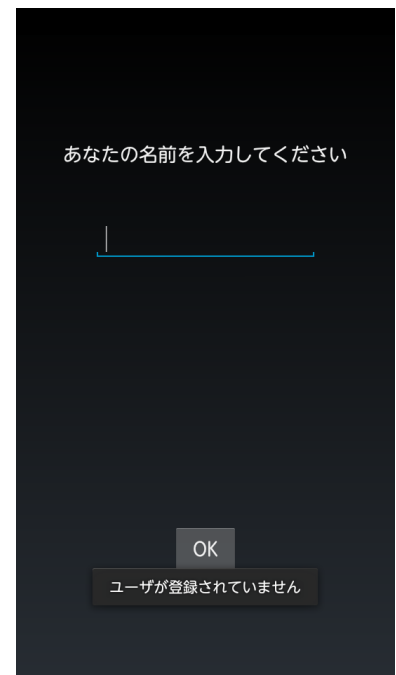


図 3.12: エラー通知

ソースコード 3.2: ユーザ確認

```

1 public class AuthNameInput extends Activity {
2     // ユーザが入力した文字列（名前）を格納する
3     public static String name;
4     ...
5     private void nameInput () {
6         final EditText nameInput = (EditText) findViewById(R.id.
            nameInputEditText);
7
8         nameInput.addTextChangedListener(new TextWatcher() {
9             ...
10            public void afterTextChanged (Editable s) {
11                // ユーザの入力した名前をnameに格納
12                if (nameInput.getText() != null) name = nameInput.getText().
                    toString().trim();
13            }
14        });
15        ...
16        // OKボタンを押した際に、次のアクティビティに移動
17        final Button ok = (Button) findViewById(R.id.okButton);
18
19        ok.setOnClickListener(new View.OnClickListener() {
20            @Override
21            public void onClick (View v) {

```



```
22 // 指定したユーザが存在するかどうかを確認する .
23 if (AuthNameInput.this.checkUserExists()) {
24     AuthNameInput.this.moveActivity("com.example.motionauth", "
        com.example.motionauth.Authentication.AuthMotion", true
        );
25 }
26 else {
27     Toast.makeText(current, "ユーザが登録されていませ
        ん", Toast.LENGTH_LONG).show();
28 }
29 }
30 });
31 }
32
33 private boolean checkUserExists () {
34     Context mContext = AuthNameInput.this.getApplicationContext();
35     SharedPreferences preferences = mContext.getSharedPreferences("UserList
        ", Context.MODE_PRIVATE);
36
37     return preferences.contains(name);
38 }
39 }
```

認証試験モードでは新規登録モードと異なり、1 回分のモーションデータ取得を行う。モーションデータの取得後、新規登録モードにおいて登録された増幅量を元に取得したデータに対して増幅処理を行う。そして、フーリエ変換を用いたローパスフィルタ処理を行い、新規登録モードにおいて登録されたデータとの相関係数を算出し個人認証を行う。個人認証に成功すれば、図 3.13 のような認証成功ダイアログを表示する。このダイアログの OK ボタンを押すことで、起動画面へと移動する。個人認証に失敗すれば、図 3.14 のような認証失敗ダイアログを表示する。このダイアログの OK ボタンを押すことでプログラム内部のモーションデータ取得カウンタが初期化され、再度個人認証を行えるようにする。

3.2.3 データ閲覧モード

データ閲覧モードでは、新規登録モードにおいて登録されたユーザ名とそれぞれのユーザが登録したモーションデータを閲覧することが出来る。このモードを選択すると、まず図 3.15 のようなユーザ名の一覧が表示される。ここでデータを閲覧したいユーザ名を選択することで、図 3.16 のようにモーションを調べることが出来る。

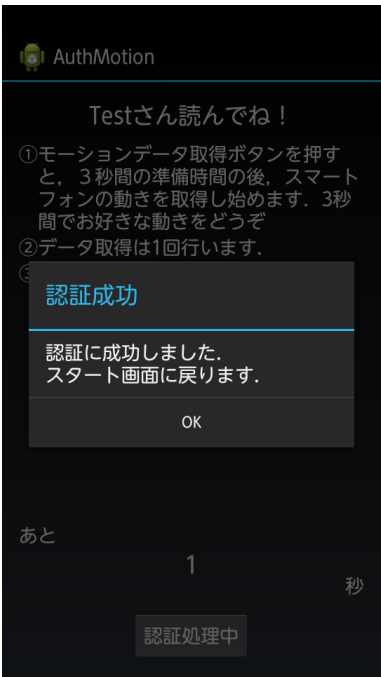


図 3.13: 認証成功ダイアログ

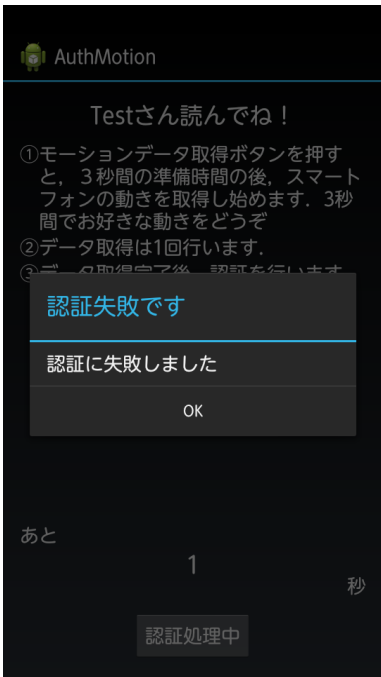


図 3.14: 認証失敗ダイアログ

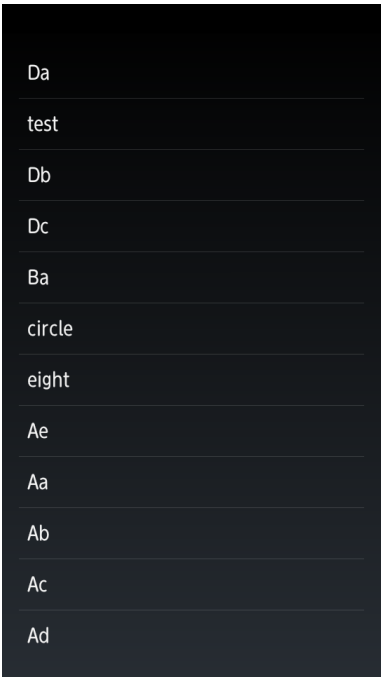


図 3.15: ユーザ名一覧画面

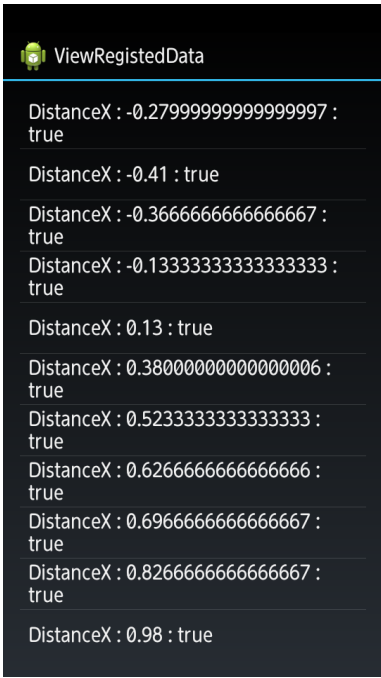


図 3.16: モーションデータ一覧画面

3.3 先行研究からの改善点

先行研究において指摘されていた、手首のスナップを用いるような比較的動きの小さなモーションにおいて、認証率が低く出てしまうという課題を解決するために導入した機能をここに挙げる。

3.3.1 モーションデータの増幅機能

手首を中心とするような動きの小さいモーションにおける個人認証成功率を向上させるために、モーションデータの増幅機能を実装した。これにより、比較的動きの小さなモーションであってもデータを増幅して用いることができ、比較的動きの大きなモーションと比べて遜色なく用いることが出来るようになった。この処理をソースコード 3.3 に示す。

ソースコード 3.3: モーションデータ増幅機能

```
1 public double[][][] Amplify (double[][][] data, double ampValue) {
2     if (ampValue != 0.0) {
3         for (int i = 0; i < data.length; i++) {
4             for (int j = 0; j < data[i].length; j++) {
5                 for (int k = 0; k < data[i][j].length; k++) {
6                     data[i][j][k] *= ampValue;
7                 }
8             }
9         }
10    }
11    return data;
12 }
```

この増幅機能は、事前にデータの最大値と最小値の差を取り、得られた値があらかじめ設定された閾値を下回った場合にのみ機能するようにしている。この処理をソースコード 3.4 に示す。

ソースコード 3.4: データレンジチェック

```
1 private boolean isRangeCheck = false;
2
3 public boolean CheckValueRange (double[][][] data, double checkRangeValue) {
4     double[][] max = new double[data.length][data[0].length];
5     double[][] min = new double[data.length][data[0].length];
6
7     for (int i = 0; i < data.length; i++) {
8         for (int j = 0; j < data[i].length; j++) {
9             max[i][j] = 0;
10            min[i][j] = 0;
11        }
12    }
```

```
13
14     double range;
15     for (int i = 0; i < data.length; i++) {
16         for (int j = 0; j < data[i].length; j++) {
17             for (int k = 0; k < data[i][j].length; k++) {
18                 if (data[i][j][k] > max[i][j]) {
19                     max[i][j] = data[i][j][k];
20                 }
21                 else if (data[i][j][k] < min[i][j]) {
22                     min[i][j] = data[i][j][k];
23                 }
24             }
25         }
26     }
27
28     for (int i = 0; i < max.length; i++) {
29         for (int j = 0; j < max[i].length; j++) {
30             range = max[i][j] - min[i][j];
31             if (range < checkRangeValue) isRangeCheck = true;
32         }
33     }
34
35     return isRangeCheck;
36 }
```

データをどれだけ増幅させるかを決める値やデータの最大値と最小値の差がどれだけあれば増幅を行うかを決める閾値に関しては、新規登録モードにおいてメニューキーを押すことで表示される図 3.17 のようなメニュー内の増幅器設定を選択することで表示される、図 3.18 のような設定ダイアログより変更することが出来る。

増幅器にかける前のデータをグラフ化したものを図 3.19 に、かけた後のデータをグラフ化したものを図 3.20 に示す。

3.3.2 フーリエ変換を用いたローパスフィルタ

モーションデータを取得している際の細かな手の震えなどによるデータに対する影響を取り除き、モーションデータとしての純度を高めるために、フーリエ変換を用いたローパスフィルタ処理を行っている。

フーリエ変換を用いて時間軸で表されるモーションデータを周波数領域に変換することで、モーション中の細かな手の震えなどのデータが高周波成分として現れる。そしてこの高周波成分を取り除いた

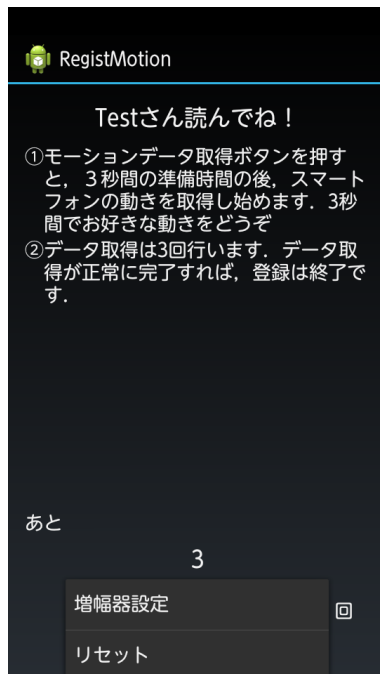


図 3.17: 新規登録モードメニュー画面

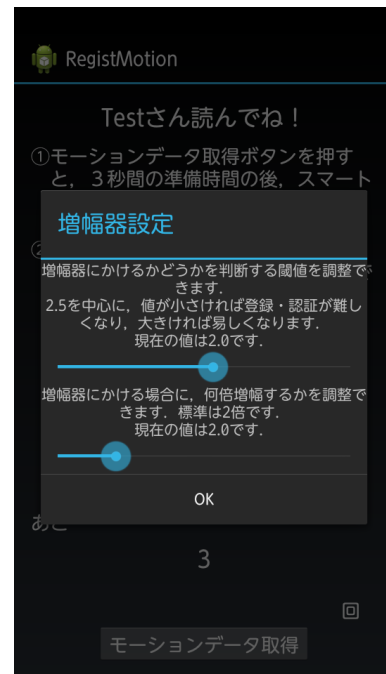


図 3.18: 増幅器設定ダイアログ

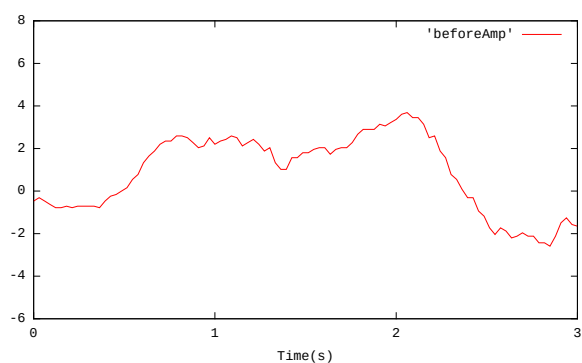


図 3.19: 増幅前のデータ

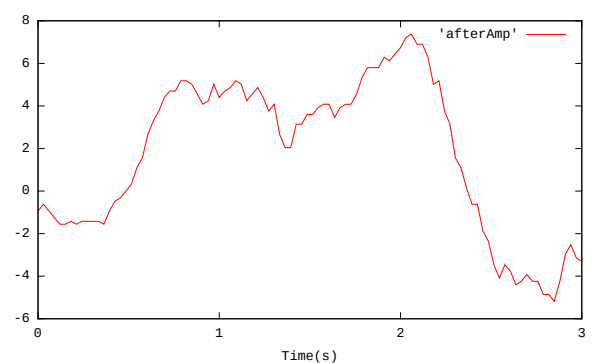


図 3.20: 増幅後のデータ

上で元の時間軸のデータに戻すローパスフィルタ処理を行うことで、細かな手の震えなどによるデータに対する影響を取り除いている。フーリエ変換を実装する際には、CERNのColt Project[3]で開発されたJavaによる科学技術計算用ライブラリであるColt[4]をマルチスレッド化したParallel Colt[5]に含まれている、JTransforms[6]を用いて実装している。この処理をソースコード3.5に示す。

ソースコード 3.5: ローパスフィルタ処理

```
1 public double[][][] LowpassFilter (double[][][] data, String dataName) {
2     DoubleFFT_1D realfft = new DoubleFFT_1D(data[0][0].length);
3
4     // フーリエ変換を実行
5     for (double[][] i : data) {
6         for (double[] j : i) {
7             realfft.realForward(j);
8         }
9     }
10
11     // 実数部, 虚数部それぞれを入れる配列
12     double[][][] real = new double[data.length][data[0].length][data[0][0].length];
13     double[][][] imaginary = new double[data.length][data[0].length][data[0][0].length];
14
15     int countReal = 0;
16     int countImaginary = 0;
17
18     // 実数部と虚数部に分解
19     for (int i = 0; i < data.length; i++) {
20         for (int j = 0; j < data[i].length; j++) {
21             for (int k = 0; k < data[i][j].length; k++) {
22                 if (k % 2 == 0) {
23                     real[i][j][countReal] = data[i][j][k];
24                     countReal++;
25                     if (countReal == 99) countReal = 0;
26                 }
27                 else {
28                     imaginary[i][j][countImaginary] = data[i][j][k];
29                     countImaginary++;
30                     if (countImaginary == 99) countImaginary = 0;
31                 }
32             }
33         }
34     }
```

```

34     }
35
36     // ローパスフィルタ処理
37     for (int i = 0; i < data.length; i++) {
38         for (int j = 0; j < data[i].length; j++) {
39             for (int k = 0; k < data[i][j].length; k++) {
40                 if (k > 30) data[i][j][k] = 0;
41             }
42         }
43     }
44
45     // 逆フーリエ変換を実行
46     for (double[][] i : data) {
47         for (double[] j : i) {
48             realfft.realInverse(j, true);
49         }
50     }
51
52     return data;
53 }

```

ローパスフィルタにかける前のデータをグラフ化したものを図 3.21 に、かけた後のデータをグラフ化したものを図 3.22 に示す。

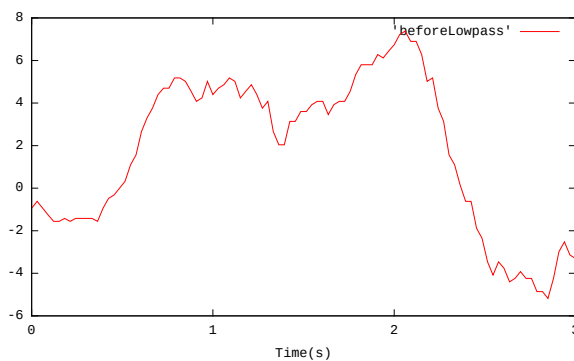


図 3.21: ローパスフィルタ前のデータ

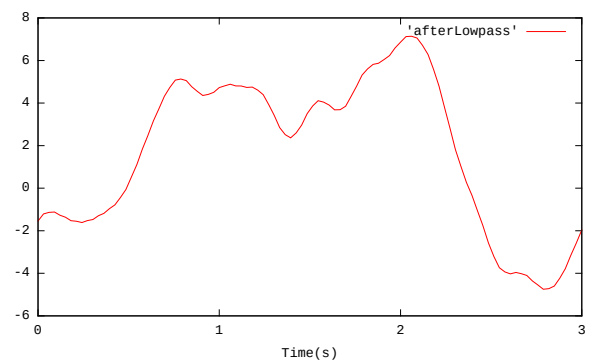


図 3.22: ローパスフィルタ後のデータ

3.3.3 モーション取得時のズレを修正する機能

新規登録モードにおいてモーションを登録する際には3回分のモーションデータの入力が必要になるが、この回数ごとにモーションの時間的なズレが生じた場合はデータ登録や認証時に影響を与えてしまう可能性があるため、このズレを修正する処理を3回分のデータ取得後に行うようにしている。

3回分のデータ取得後、まずはこのデータ間の相関係数を算出し、回数ごとに全く別のモーションが

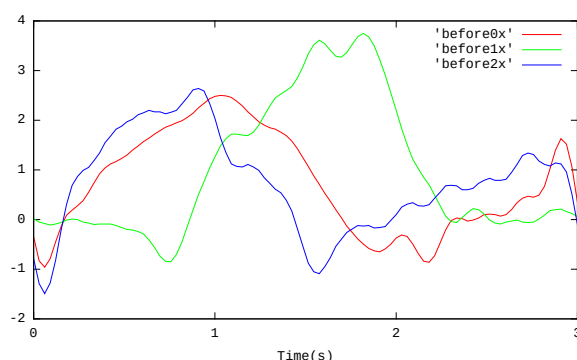


図 3.23: ズレ修正前のデータ

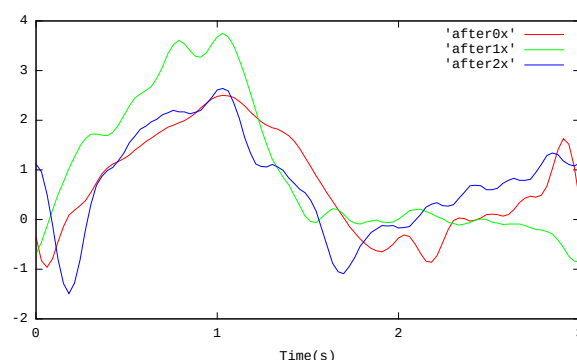


図 3.24: ズレ修正後のデータ

入力されていないかの確認を行う．そしてある程度以上の相関が見られるが，それでも相関係数が低く出てしまった場合に，ズレ修正後の相関係数を確認しつつ，データの最大値に合わせるパターン，データの最小値に合わせるパターン，データの中央値に合わせるパターンの最大三つの方法で相関係数の向上を試みる．この部分の処理を付録のソースコード A.1 に示す．

相関係数を算出した結果，ズレ修正が必要であると判断された場合，付録のソースコード A.2 に示したズレ修正アルゴリズムを用いて修正を行う．

ズレ修正を行う前のデータをグラフ化したものを図 3.23 に，修正を行った後のデータをグラフ化したものを図 3.24 に示す．

3.3.4 モーション取得時のインターバル及びヴァイブレーション機能

新規登録モード及び認証試験モードにおいてモーション取得時の時間経過が把握しやすいように，1 秒毎に端末をヴァイブレーションさせる機能を実装した．また，データ取得前のインターバルからデータ取得に移る際には通常より長めのヴァイブレーションにすることで，データ取得開始のタイミングを明確に意識できるようにした．

第4章 実験と考察

4.1 実験方法

手首のスナップを用いるような比較的小さなモーションにおける本システムの動作精度を確かめるために、新規登録および個人認証の成功率と成功時の平均試行回数を求める実験を行った。実験を行う前に任意のモーションで新規登録と個人認証を行い、個人認証に成功できた10名を被験者とした。実験対象のモーションは以下に挙げる5種類とし、手首から先のみを動かすようにしてモーションを入力した。新規登録および個人認証それぞれの試行回数は3回までとし、これで登録および認証出来なかったものは失敗とみなした。

- 円を描く
- 三拍子を振る
- 四拍子を振る
- 無限大記号を描く
- 四角形を描く

4.2 実験結果

新規登録の成功率と成功時の平均試行回数をまとめたものを表4.1に示す。

表 4.1: 新規登録の実験結果

モーション	登録成功率	平均試行回数
円を描く	100%	1.1 回
三拍子を振る	100%	1.0 回
四拍子を振る	100%	1.0 回
無限大記号を描く	100%	1.0 回
四角形を描く	100%	1.0 回

実験の結果、新規登録は被験者全員がモーションの登録に成功した。新規登録に成功した場合の試行回数は、ほぼ全員が1回のみだった。

次に，個人認証の成功率と成功時の平均試行回数をまとめたものを表 4.2 に示す．

表 4.2: 個人認証の実験結果

モーション	認証成功率	平均試行回数
円を描く	70%	1.7 回
三拍子を振る	30%	1.6 回
四拍子を振る	30%	1.0 回
無限大記号を描く	80%	1.2 回
四角形を描く	60%	1.5 回

実験の結果，個人認証の成功率に関しては高いもので 80%，低いもので 30%という結果となった．個人認証に成功した場合の全体的な平均試行回数は 1.4 回となった．

また，実験に関する詳細なデータは付録??に載せている．

4.3 考察

実験の結果から，手首から先のみを動かすようなモーションにおいて新規登録に関して失敗するということは無くなったが，個人認証に関してはモーションによって成功率が低く出てしまうものもあるという結果となった．被験者別に実験データを見てみると，認証に成功しやすい人と成功しにくい人に二分化する傾向が見られた．これについて，新規登録時と個人認証時にスマートフォンを同じように動かすという行為に馴染めたかどうかの原因の一つとして考えられる．新規登録においては，本研究のシステムにおいて新たに実装したデータ取得時の時間的なズレを修正する処理を施したことによって登録に成功した被験者が見受けられた．

モーション別に実験データを見てみると，三拍子を振るものと四拍子を振るものの認証成功率が特に低かった．これらモーションは，他のモーションに比べてモーションの馴染みが薄く，再現が難しかったのではないかと考えている．また，四拍子を振るものや四角形を描くものに関しては，モーション入力時間が 3 秒間である中でこれらモーションを入力しなければならず，入力がしづらかったのではないかと考えている．

4.4 課題

現状ではモーションの入力時間が 3 秒間と限定されてしまい，自由度が高いとはいえないため，任意の入力時間で登録や認証を行えるようにする必要がある．また，なりすましによる認証が行われた場合にどの程度耐性を有するのかの確認と，新規登録時からある程度期間を置いて個人認証を行う場合にどの程度認証に成功するのかの確認を行う必要がある．

第5章 おわりに

本研究では，モーションの振れ幅を増幅する機能やフーリエ変換を用いたローパスフィルタ，モーション取得時のズレを修正する機能を実装し，新規登録や個人認証における精度を向上することができた．しかし，モーションの入力時間が3秒間と制限されていることから，モーションの種類によっては入力がしづらいという問題がある．また，なりすましによる認証に対してどの程度の耐性を有するのかという点や，新規登録時からある程度期間を置いて個人認証を行うような場合にどの程度の割合で成功するのかといった点を検証する必要がある．さらに，モーションを入力する人によって個人認証時に成功しやすい人と成功しにくい人に二分化する傾向が見られたため，成功しにくい人のデータを精査し，どのようにカバーしていくのかを検討する必要がある．

謝辞

本研究のプログラム開発や実験，本論文の執筆にあたり，手厚い指導と様々な助言をしていただいた，関西大学総合情報学部セキュア情報システム研究室の小林孝史准教授に深く感謝いたします．また，研究テーマの選定をはじめ，日頃から有益なアドバイスを頂いた同研究室の皆様に感謝いたします．

参考文献，参考 URL 等

- [1] 坂本 翔，ユーザの直感的な入力をとらえるための 3 軸加速度センサによるジェスチャ認識の研究，2009 年度公立はこだて未来大学卒業論文．
- [2] 兎澤星伸，三軸加速度センサ及び三軸ジャイロセンサを用いた認証アプリケーションの開発，2012 年度卒業研究．
- [3] Colt Project，<https://dst.lbl.gov/ACSSoftware/colt/>，2014 年 12 月 27 日確認．
- [4] Colt，<https://github.com/carlsonp/Colt>，2014 年 12 月 27 日確認．
- [5] Parallel Colt，<https://sites.google.com/site/piotrwendykier/software/parallelcolt>，2014 年 12 月 27 日確認．
- [6] JTransforms，<https://sites.google.com/site/piotrwendykier/software/jtransforms>，2014 年 12 月 27 日確認．

付 録A プログラムより抜粋したもの

A.1 ズレ修正の判断部分

```
1 Enum.MEASURE measure = mCorrelation.measureCorrelation(distance, angle,
    averageDistance, averageAngle);
2
3 if (Enum.MEASURE.BAD == measure) {
4     // 相関係数が0.4以下
5     return false;
6 }
7 else if (Enum.MEASURE.INCORRECT == measure) {
8     // 相関係数が0.4よりも高く, 0.6以下の場合, ズレ修正を行う
9     int time = 0;
10    Enum.MODE mode = Enum.MODE.MAX;
11    Enum.TARGET target = Enum.MODE.DISTANCE;
12
13    double[][][] originalDistance = distance;
14    double[][][] originalAngle = angle;
15
16    while (true) {
17        switch (time) {
18            case 0:
19                mode = Enum.MODE.MAX;
20                target = Enum.TARGET.DISTANCE;
21                break;
22            case 1:
23                mode = Enum.MODE.MAX;
24                target = Enum.TARGET.ANGLE;
25                break;
26            case 2:
27                mode = Enum.MODE.MIN;
28                target = Enum.TARGET.DISTANCE;
29                break;
30            case 3:
```

```

31         mode = Enum.MODE.MIN;
32         target = Enum.TARGET.ANGLE;
33         break;
34     case 4:
35         mode = Enum.MODE.MEDIAN;
36         target = Enum.TARGET.DISTANCE;
37         break;
38     case 5:
39         mode = Enum.MODE.MEDIAN;
40         target = Enum.TARGET.ANGLE;
41         break;
42     }
43
44     double[][][][] deviatedValue = mCorrectDeviation.correctDeviation(
45         originalDistance, originalAngle, mode, target);
46
47     for (int i = 0; i < 3; i++) {
48         for (int j = 0; j < 3; j++) {
49             for (int k = 0; k < 100; k++) {
50                 distance[i][j][k] = deviatedValue[0][i][j][k];
51                 angle[i][j][k] = deviatedValue[1][i][j][k];
52             }
53         }
54     }
55
56     for (int i = 0; i < 3; i++) {
57         for (int j = 0; j < 100; j++) {
58             averageDistance[i][j] = (distance[0][i][j] + distance[1][i][j]
59                 + distance[2][i][j]) / 3;
60             averageAngle[i][j] = (angle[0][i][j] + angle[1][i][j] + angle
61                 [2][i][j]) / 3;
62         }
63     }
64
65     Enum.MEASURE tmp = mCorrelation.measureCorrelation(distance, angle,
66         averageDistance, averageAngle);
67
68     if (tmp == Enum.MEASURE.PERFECT || tmp == Enum.MEASURE.CORRECT) {
69         break;
70     }
71     else if (time == 2) {

```

```

68         // 相関係数が低いまま
69         distance = originalDistance;
70         angle = originalAngle;
71         break;
72     }
73
74     time++;
75 }
76 }
77 else if (Enum.MEASURE.PERFECT == measure || Enum.MEASURE.CORRECT == measure) {
78     // 何もしない
79 }
80 else {
81     return false;
82 }

```

A.2 ズレ修正アルゴリズム

```

1 public double[][][] correctDeviation (double[][][] data, Enum.MODE mode) {
2     double[][][] newData = new double[3][3][100];
3
4     double value[][] = new double[3][3];
5     int count[][] = new int[3][3];
6
7     // 変数にXYZそれぞれの一個目の値を放り込む
8     for (int i = 0; i < 3; i++) {
9         for (int j = 0; j < 3; j++) {
10             value[i][j] = data[i][j][0];
11         }
12     }
13
14     // 代表値が出ている場所を取得する
15     switch (mode) {
16         case MAX:
17             for (int i = 0; i < 3; i++) {
18                 for (int j = 0; j < 3; j++) {
19                     for (int k = 0; k < 100; k++) {
20                         if (value[i][j] < data[i][j][k]) {
21                             value[i][j] = data[i][j][k];
22                             count[i][j] = k;
23                         }

```



```
24         }
25     }
26 }
27 break;
28 case MIN:
29     for (int i = 0; i < 3; i++) {
30         for (int j = 0; j < 3; j++) {
31             for (int k = 0; k < 100; k++) {
32                 if (value[i][j] > data[i][j][k]) {
33                     value[i][j] = data[i][j][k];
34                     count[i][j] = k;
35                 }
36             }
37         }
38     }
39     break;
40 case MEDIAN:
41     // キーが自動ソートされる
42     // TreeMapを用いる。データと順番を紐付けしたものを作成し、中央値の初期の順番
43
44     for (int i = 0; i < 3; i++) {
45         for (int j = 0; j < 3; j++) {
46             TreeMap<Double, Integer> treeMap = new TreeMap<>();
47
48             for (int k = 0; k < 100; k++) {
49                 treeMap.put(data[i][j][k], k);
50             }
51
52             int loopCount = 0;
53             for (Integer initCount : treeMap.values()) {
54                 if (loopCount == 49) {
55                     count[i][j] = initCount;
56                 }
57                 loopCount++;
58             }
59         }
60     }
61     break;
```

```
62 // 1回目のデータの代表値が出た場所と、2回目・3回目のデータの代表値が出た場  
    所の差をとる  
63 // とったら、その差だけデータをずらす（ずらしてはみ出たデータは空いたところ  
    に入れる）  
64  
65 int lagData[][] = new int[2][3];  
66  
67 // どれだけズレているかを計算する  
68 for (int i = 0; i < 3; i++) {  
69     lagData[0][i] = count[0][i] - count[1][i];  
70     lagData[1][i] = count[0][i] - count[2][i];  
71 }  
72  
73 // 1回目のデータに関しては基準となるデータなのでそのまま入れる  
74 for (int i = 0; i < 3; i++) {  
75     for (int j = 0; j < 100; j++) {  
76         newData[0][i][j] = data[0][i][j];  
77     }  
78 }  
79  
80 // 実際にリストの要素をずらしていく（ずらすのは、二回目と三回目のデータのみ  
    ）  
81 for (int i = 1; i < 3; i++) {  
82     for (int j = 0; j < 3; j++) {  
83         ArrayList<Double> temp = new ArrayList<>();  
84  
85         for (int k = 0; k < data[i][j].length; k++) {  
86             temp.add(data[i][j][k]);  
87         }  
88  
89         Collections.rotate(temp, lagData[i - 1][j]);  
90  
91         for (int k = 0; k < data[i][j].length; k++) {  
92             newData[i][j][k] = temp.get(k);  
93         }  
94     }  
95 }  
96  
97 return newData;  
98 }
```

付 録 B 実験結果詳細

被験者	モーション	試行回数-新規登録	試行回数-個人認証
A	円を描く	1 回目で成功	7 回目まで失敗
	三拍子を描く	1 回目で成功	5 回目で成功
	四拍子を描く	1 回目で成功	4 回目で成功
	無限大記号を描く	1 回目で成功	1 回目で成功
	四角形を描く	1 回目で成功	1 回目で成功
B	円を描く	1 回目で成功	7 回目まで失敗
	三拍子を描く	1 回目で成功	3 回目まで失敗
	四拍子を描く	1 回目で成功	4 回目まで失敗
	無限大記号を描く	1 回目で成功	7 回目まで失敗
	四角形を描く	1 回目で成功	4 回目まで失敗
C	円を描く	1 回目で成功	3 回目で成功
	三拍子を描く	1 回目で成功	3 回目まで失敗
	四拍子を描く	1 回目で成功	5 回目まで失敗
	無限大記号を描く	1 回目で成功	4 回目まで失敗
	四角形を描く	1 回目で成功	5 回目まで失敗
D	円を描く	1 回目で成功	2 回目で成功
	三拍子を描く	1 回目で成功	1 回目で成功
	四拍子を描く	1 回目で成功	1 回目で成功
	無限大記号を描く	1 回目で成功	1 回目で成功
	四角形を描く	1 回目で成功	1 回目で成功
E	テスト段階で認証できず		
	円を描く	1 回目で成功	1 回目で成功
	三拍子を描く	1 回目で成功	7 回目まで失敗

被験者	モーション	試行回数-新規登録	試行回数-個人認証
F	四拍子を描く	1 回目で成功	1 回目で成功
	無限大記号を描く	1 回目で成功	2 回目で成功
	四角形を描く	1 回目で成功	1 回目で成功
G	円を描く	1 回目で成功	2 回目で成功
	三拍子を描く	1 回目で成功	3 回目まで失敗
	四拍子を描く	1 回目で成功	3 回目まで失敗
	無限大記号を描く	1 回目で成功	1 回目で成功
	四角形を描く	1 回目で成功	3 回目で成功
H	円を描く	1 回目で成功	1 回目で成功
	三拍子を描く	1 回目で成功	1 回目で成功
	四拍子を描く	1 回目で成功	1 回目で成功
	無限大記号を描く	1 回目で成功	1 回目で成功
	四角形を描く	1 回目で成功	1 回目で成功
I	円を描く	2 回目で成功	1 回目で成功
	三拍子を描く	1 回目で成功	3 回目まで失敗
	四拍子を描く	1 回目で成功	3 回目まで失敗
	無限大記号を描く	1 回目で成功	2 回目で成功
	四角形を描く	1 回目で成功	3 回目まで失敗
J	円を描く	1 回目で成功	2 回目で成功
	三拍子を描く	1 回目で成功	3 回目で成功
	四拍子を描く	1 回目で成功	3 回目まで失敗
	無限大記号を描く	1 回目で成功	1 回目で成功
	四角形を描く	1 回目で成功	2 回目で成功
K	円を描く	1 回目で成功	3 回目まで失敗
	三拍子を描く	1 回目で成功	3 回目まで失敗
	四拍子を描く	1 回目で成功	3 回目まで失敗
	無限大記号を描く	1 回目で成功	1 回目で成功
	四角形を描く	1 回目で成功	3 回目まで失敗

付 録C プログラム本体

C.1 src/com/example/motionauth/Start.java

```
1 package com.example.motionauth;
2
3 import android.app.Activity;
4 import android.app.AlertDialog;
5 import android.content.DialogInterface;
6 import android.content.Intent;
7 import android.os.Bundle;
8 import android.os.Handler;
9 import android.os.Message;
10 import android.util.Log;
11 import android.view.KeyEvent;
12 import android.view.View;
13 import android.view.Window;
14 import android.widget.Button;
15 import com.example.motionauth.Utility.LogUtil;
16
17
18 /**
19  * アプリを起動した際に最初に表示されるアクティビティ
20  * モード選択を行う
21  *
22  * @author Kensuke Kousaka
23  */
24 public class Start extends Activity {
25     private final static int POSITIVE = 1;
26     private final static int NEUTRAL = 2;
27     private final static int NEGATIVE = 3;
28
29     private final static int DOUBLE = 2;
30     private final static int TRIPLE = 3;
31 }
```

```
32 private Handler handler;
33
34 @Override
35 protected void onCreate (Bundle savedInstanceState) {
36     super.onCreate(savedInstanceState);
37
38     // 設定ファイルのフラグを読み取ってログ出力を切り替える
39     boolean isShowLog = getResources().getBoolean(R.bool.isShowLog);
40     LogUtil.setShowLog(isShowLog);
41
42     LogUtil.log(Log.INFO);
43
44     // タイトルバーの非表示
45     requestWindowFeature(Window.FEATURE_NO_TITLE);
46     setContentView(R.layout.activity_start);
47
48     selectMode();
49 }
50
51
52 /**
53  * モード選択
54  */
55 private void selectMode () {
56     LogUtil.log(Log.INFO);
57
58     Button startBtn = (Button) findViewById(R.id.start);
59
60     startBtn.setOnClickListener(new View.OnClickListener() {
61         @Override
62         public void onClick (View v) {
63             LogUtil.log(Log.DEBUG, "Click start button");
64
65             String[] btnMsg = {"データ閲覧モード", "認証試験モード", "新規登録モード"};
66             Start.this.alertDialog(TRIPLE, btnMsg, "モード選択", "モードを選択してください");
67             handler = new Handler() {
68                 public void handleMessage (Message msg) {
69                     if (msg.arg1 == POSITIVE) {
70                         LogUtil.log(Log.DEBUG, "POSITIVE");
71                         // 登録者一覧モード
```

```
72         moveActivity("com.example.motionauth", "com.example.motionauth.
           ViewDataList.RegistrantList", true);
73     }
74     else if (msg.arg1 == NEUTRAL) {
75         LogUtil.log(Log.DEBUG, "NEUTRAL");
76         // 認証試験モード
77         moveActivity("com.example.motionauth", "com.example.motionauth.
           Authentication.AuthNameInput", true);
78     }
79     else if (msg.arg1 == NEGATIVE) {
80         LogUtil.log(Log.DEBUG, "NEGATIVE");
81         // 新規登録モード
82         moveActivity("com.example.motionauth", "com.example.motionauth.
           Registration.RegistNameInput", true);
83     }
84 }
85 };
86 }
87 });
88 }
89
90
91 /**
92  * アラートダイアログ作成
93  *
94  * @param choiceNum 2択か3択か
95  * @param btnMsg     選択肢ボタンの文字列
96  * @param title      ダイアログのタイトル
97  * @param msg        ダイアログの説明
98  */
99 private void alertDialog (int choiceNum, String[] btnMsg, String title ,
       String msg) {
100     LogUtil.log(Log.INFO);
101
102     if (choiceNum == DOUBLE) {
103         LogUtil.log(Log.DEBUG, "DOUBLE");
104
105         AlertDialog.Builder alert = new AlertDialog.Builder(this);
106         alert.setOnKeyListener(new DialogInterface.OnKeyListener() {
107             @Override
```

```
108     public boolean onKeyDown (DialogInterface dialog, int keyCode, KeyEvent
        event) {
109         // アラート画面に特定のキー動作をかませる
110         if (keyCode == KeyEvent.KEYCODE_BACK) {
111             // Backキーが押された場合
112             // ダイアログを閉じて、アクティビティを閉じる
113             dialog.dismiss();
114             Start.this.finish();
115
116             return true;
117         }
118         return false;
119     }
120 });
121
122 // ダイアログ外をタッチしてもダイアログが閉じないようにする
123 alert.setCancelable(false);
124
125 alert.setTitle(title);
126 alert.setMessage(msg);
127
128 // PositiveButtonにより、ダイアログの左側に配置される
129 alert.setPositiveButton(btnMsg[0], new DialogInterface.OnClickListener()
    {
130     @Override
131     public void onClick (DialogInterface dialog, int which) {
132         Message msg1 = new Message();
133         msg1.arg1 = POSITIVE;
134
135         handler.sendMessage(msg1);
136     }
137 });
138
139 alert.setNegativeButton(btnMsg[1], new DialogInterface.OnClickListener()
    {
140     @Override
141     public void onClick (DialogInterface dialog, int which) {
142         Message msg1 = new Message();
143         msg1.arg1 = NEGATIVE;
144
145         handler.sendMessage(msg1);
```



```
146     }
147     });
148
149     // ダイアログを表示する
150     alert.show();
151 }
152 else if (choiceNum == TRIPLE) {
153     LogUtil.log(Log.DEBUG, "TRIPLE");
154
155     AlertDialog.Builder alert = new AlertDialog.Builder(this);
156     alert.setOnKeyListener(new DialogInterface.OnKeyListener() {
157         @Override
158         public boolean onKey (DialogInterface dialog, int keyCode, KeyEvent
159             event) {
160             // アラート画面に特定のキー動作をかませる
161             if (keyCode == KeyEvent.KEYCODE.BACK) {
162                 // Backキーを押した場合
163                 // ダイアログを閉じて、アクティビティを閉じる
164                 dialog.dismiss();
165                 Start.this.finish();
166
167                 return true;
168             }
169             return false;
170         }
171     });
172
173     // ダイアログ外をタッチしてもダイアログを閉じないようにする
174     alert.setCancelable(false);
175
176     alert.setTitle(title);
177     alert.setMessage(msg);
178
179     // PositiveButtonにより、ダイアログの左側に配置される
180     alert.setPositiveButton(btnMsg[0], new DialogInterface.OnClickListener() {
181         @Override
182         public void onClick (DialogInterface dialog, int which) {
183             Message msg1 = new Message();
184             msg1.arg1 = POSITIVE;
```

```
185         handler.sendMessage(msg1);
186     }
187 });
188
189 alert.setNeutralButton(btnMsg[1], new DialogInterface.OnClickListener() {
190     @Override
191     public void onClick (DialogInterface dialog, int which) {
192         Message msg1 = new Message();
193         msg1.arg1 = NEUTRAL;
194
195         handler.sendMessage(msg1);
196     }
197 });
198
199 alert.setNegativeButton(btnMsg[2], new DialogInterface.OnClickListener()
200     {
201     @Override
202     public void onClick (DialogInterface dialog, int which) {
203         Message msg1 = new Message();
204         msg1.arg1 = NEGATIVE;
205
206         handler.sendMessage(msg1);
207     }
208 });
209
210 // ダイアログを表示する
211 alert.show();
212 }
213
214
215 /**
216  * アクティビティを移動する
217  *
218  * @param pkgName 移動先のパッケージ名
219  * @param actName 移動先のアクティビティ名
220  * @param flg      戻るキーを押した際にこのアクティビティを表示させるかどうか
221  */
222 private void moveActivity (String pkgName, String actName, boolean flg) {
223     LogUtil.log(Log.INFO);
224 }
```

```
225     Intent intent = new Intent();
226
227     intent.setClassName(pkgName, actName);
228
229     if (flg) intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.
        FLAG_ACTIVITY_NEW_TASK);
230
231     startActivityForResult(intent, 0);
232 }
233 }
```

C.2 src/com/example/motionauth/Registration/RegistNameInput.java

```
1 package com.example.motionauth.Registration;
2
3 import android.app.Activity;
4 import android.content.Context;
5 import android.content.Intent;
6 import android.os.Bundle;
7 import android.text.Editable;
8 import android.text.TextWatcher;
9 import android.util.Log;
10 import android.view.KeyEvent;
11 import android.view.View;
12 import android.view.Window;
13 import android.view.inputmethod.InputMethodManager;
14 import android.widget.Button;
15 import android.widget.EditText;
16 import android.widget.Toast;
17 import com.example.motionauth.R;
18 import com.example.motionauth.Utility.LogUtil;
19
20
21 /**
22  * ユーザに名前を入力させる
23  *
24  * @author Kensuke Kousaka
25  */
26 public class RegistNameInput extends Activity {
27     // ユーザが入力した文字列（名前）を格納する
28     public static String name;
```

```
29
30
31 @Override
32 protected void onCreate (Bundle savedInstanceState) {
33     super.onCreate(savedInstanceState);
34
35     LogUtil.log(Log.INFO);
36
37     // タイトルバーの非表示
38     requestWindowFeature(Window.FEATURE_NO_TITLE);
39     setContentView(R.layout.activity_regist_name_input);
40
41     name = "";
42
43     nameInput();
44 }
45
46
47 /**
48  * ユーザの名前入力を受け付ける処理
49  */
50 private void nameInput () {
51     LogUtil.log(Log.INFO);
52
53     final EditText nameInput = (EditText) findViewById(R.id.nameInputEditText);
54
55     nameInput.addTextChangedListener(new TextWatcher() {
56         // 変更前
57         public void beforeTextChanged (CharSequence s, int start, int count, int
58             after) {
59
60         // 変更直前
61         public void onTextChanged (CharSequence s, int start, int before, int
62             count) {
63
64         // 変更後
65         public void afterTextChanged (Editable s) {
66             // ユーザの入力した名前をnameに格納
```

```
67         if (nameInput.getText() != null) name = nameInput.getText().toString().
           trim();
68     }
69 });
70
71 nameInput.setOnKeyListener(new View.OnKeyListener() {
72     @Override
73     public boolean onKey (View v, int keyCode, KeyEvent event) {
74         if (event.getAction() == KeyEvent.ACTION_DOWN && keyCode == KeyEvent.
           KEYCODE_ENTER) {
75             // ソフトウェアキーボードの
              Enterキーを押した時、ソフトウェアキーボードを閉じる
76             InputMethodManager inputMethodManager = (InputMethodManager)
              RegisterNameInput.this.getSystemService(Context.INPUT_METHOD_SERVICE
              );
77             inputMethodManager.hideSoftInputFromWindow(v.getWindowToken(), 0);
78
79             return true;
80         }
81         return false;
82     }
83 });
84
85 // OKボタンを押した時に、次のアクティビティに移動
86 final Button ok = (Button) findViewById(R.id.okButton);
87
88 ok.setOnClickListener(new View.OnClickListener() {
89     @Override
90     public void onClick (View v) {
91         LogUtil.log(Log.DEBUG, "Click ok button");
92         // nameが入力されているかの確認
93         if (name.length() == 0) {
94             Toast.makeText(RegisterNameInput.this, "名前が入力されていませ
              へん", Toast.LENGTH_LONG).show();
95         }
96         else {
97             RegisterNameInput.this.moveActivity("com.example.motionauth", "com.
              example.motionauth.Registration.RegistMotion", true);
98         }
99     }
100 });
101 }
```

```
102
103
104  /**
105   * アクティビティを移動する
106   *
107   * @param pkgName 移動先のパッケージ名
108   * @param actName 移動先のアクティビティ名
109   * @param flg      戻るキーを押した際にこのアクティビティを表示させるかどうか
110   */
111  private void moveActivity (String pkgName, String actName, boolean flg) {
112      LogUtil.log(Log.INFO);
113      Intent intent = new Intent();
114
115      intent.setClassName(pkgName, actName);
116
117      if (flg) intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.
118          FLAG_ACTIVITY_NEW_TASK);
119
120      startActivityForResult(intent, 0);
121  }
122 }
```

C.3 src/com/example/motionauth/Registration/RegistMotion.java

```
1  package com.example.motionauth.Registration;
2
3  import android.app.Activity;
4  import android.app.AlertDialog;
5  import android.app.ProgressDialog;
6  import android.content.Context;
7  import android.content.DialogInterface;
8  import android.content.Intent;
9  import android.hardware.Sensor;
10 import android.hardware.SensorEvent;
11 import android.hardware.SensorEventListener;
12 import android.hardware.SensorManager;
13 import android.os.Bundle;
14 import android.os.Handler;
15 import android.os.Message;
16 import android.os.Vibrator;
```

```
17 import android.util.Log;
18 import android.view.*;
19 import android.widget.Button;
20 import android.widget.SeekBar;
21 import android.widget.TextView;
22 import com.example.motionauth.Lowpass.Fourier;
23 import com.example.motionauth.Processing.*;
24 import com.example.motionauth.R;
25 import com.example.motionauth.Utility.Enum;
26 import com.example.motionauth.Utility.LogUtil;
27 import com.example.motionauth.Utility.ManageData;
28
29
30 /**
31  * モーションを新規登録する
32  *
33  * @author Kensuke Kousaka
34  */
35 public class RegistMotion extends Activity implements SensorEventListener,
    Runnable {
36     private static final int VIBRATOR_SHORT = 25;
37     private static final int VIBRATOR_NORMAL = 50;
38     private static final int VIBRATOR_LONG = 100;
39
40     private static final int PREPARATION = 1;
41     private static final int GET_MOTION = 2;
42
43     private static final int PREPARATION_INTERVAL = 1000;
44     private static final int GET_MOTION_INTERVAL = 30;
45
46     private static final int FINISH = 5;
47
48     private SensorManager mSensorManager;
49     private Sensor mAccelerometerSensor;
50     private Sensor mGyroscopeSensor;
51
52     private Vibrator mVibrator;
53
54     private TextView secondTv;
55     private TextView countSecondTv;
56     private Button getMotionBtn;
```

```
57
58 private Fourier      mFourier      = new Fourier();
59 private Formatter    mFormatter    = new Formatter();
60 private Calc         mCalc         = new Calc();
61 private Amplifier    mAmplifier    = new Amplifier();
62 private ManageData   mManageData   = new ManageData();
63 private Correlation  mCorrelation  = new Correlation();
64 private CorrectDeviation mCorrectDeviation = new CorrectDeviation();
65
66 private int dataCount = 0;
67 private int getCount  = 0;
68 private int prepareCount = 0;
69
70 private boolean isGetMotionBtnClickable = true;
71
72 // モーションの生データ
73 private float[] vAccel;
74 private float[] vGyro;
75
76 private float[][][] accelFloat = new float[3][3][100];
77 private float[][][] gyroFloat = new float[3][3][100];
78
79 private double[][][] distance      = new double[3][3][100];
80 private double[][][] angle        = new double[3][3][100];
81 private double[][] averageDistance = new double[3][100];
82 private double[][] averageAngle   = new double[3][100];
83
84 private boolean resultCalc = false;
85 private boolean resultCorrelation = false;
86
87 private ProgressDialog progressDialog;
88 private double checkRangeValue = 2.0;
89 private double ampValue = 2.0;
90
91 private boolean isMenuClickable = true;
92
93
94 @Override
95 protected void onCreate (Bundle savedInstanceState) {
96     super.onCreate(savedInstanceState);
97     LogUtil.log(Log.INFO);
```



```
98
99     setContentView(R.layout.activity_regist_motion);
100
101     registMotion();
102 }
103
104
105 /**
106  * モーション登録画面にイベントリスナ等を設定する
107  */
108 private void registMotion () {
109     LogUtil.log(Log.INFO);
110
111     // センササービス, 各種センサを取得する
112     mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
113     mAccelerometerSensor = mSensorManager.getDefaultSensor(Sensor.
114         TYPE_ACCELEROMETER);
115     mGyroscopeSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
116
117     mVibrator = (Vibrator) getSystemService(VIBRATOR.SERVICE);
118
119     TextView nameTv = (TextView) findViewById(R.id.textView2);
120     secondTv = (TextView) findViewById(R.id.secondTextView);
121     countSecondTv = (TextView) findViewById(R.id.textView4);
122     getMotionBtn = (Button) findViewById(R.id.button1);
123
124     nameTv.setText(RegistNameInput.name + "さん読んでね！");
125
126     getMotionBtn.setOnClickListener(new View.OnClickListener() {
127         @Override
128         public void onClick (View v) {
129             LogUtil.log(Log.DEBUG, "Click get motion button");
130             if (isGetMotionBtnClickable) {
131                 isGetMotionBtnClickable = false;
132
133                 // ボタンをクリックできないようにする
134                 v.setClickable(false);
135
136                 isMenuClickable = false;
137
138                 getMotionBtn.setText("インターバル中");
```



```
175         secondTv.setText("START");
176         mVibrator.vibrate(VIBRATOR_LONG);
177
178         // GET_MOTIONメッセージを添えて, timeHandlerを呼び出す
179         timeHandler.sendEmptyMessage(GET_MOTION);
180         getMotionBtn.setText("取得中");
181         break;
182     }
183
184     prepareCount++;
185 }
186 else if (msg.what == GET_MOTION && !isGetMotionBtnClickable) {
187     if (dataCount < 100 && getCount >= 0 && getCount < 3) {
188         // 取得した値を, 0.03秒ごとに配列に入れる
189         for (int i = 0; i < 3; i++) {
190             accelFloat[getCount][i][dataCount] = vAccel[i];
191             gyroFloat[getCount][i][dataCount] = vGyro[i];
192         }
193
194         dataCount++;
195
196         switch (dataCount) {
197             case 1:
198                 secondTv.setText("3");
199                 mVibrator.vibrate(VIBRATOR_NORMAL);
200                 break;
201             case 33:
202                 secondTv.setText("2");
203                 mVibrator.vibrate(VIBRATOR_NORMAL);
204                 break;
205             case 66:
206                 secondTv.setText("1");
207                 mVibrator.vibrate(VIBRATOR_NORMAL);
208                 break;
209         }
210
211         timeHandler.sendEmptyMessageDelayed(GET_MOTION, GET_MOTION_INTERVAL);
212     }
213     else if (dataCount >= 100 && getCount >= 0 && getCount < 4) {
214         // 取得完了
215         mVibrator.vibrate(VIBRATOR_LONG);
```

```
216         isGetMotionBtnClickable = true;
217         isMenuClickable = true;
218         getCount++;
219         countSecondTv.setText("回");
220         getMotionBtn.setText("モーションデータ取得");
221
222         dataCount = 0;
223         prepareCount = 0;
224
225         switch (getCount) {
226             case 1:
227                 secondTv.setText("2");
228                 getMotionBtn.setClickable(true);
229                 break;
230             case 2:
231                 secondTv.setText("1");
232                 getMotionBtn.setClickable(true);
233                 break;
234             case 3:
235                 finishGetMotion();
236                 break;
237         }
238     }
239     else {
240         super.dispatchMessage(msg);
241     }
242 }
243 }
244 };
245
246
247 @Override
248 public void onSensorChanged (SensorEvent event) {
249     if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) vAccel = event.
        values.clone();
250     if (event.sensor.getType() == Sensor.TYPE_GYROSCOPE) vGyro = event.values.
        clone();
251 }
252
253 private void finishGetMotion () {
254     if (getMotionBtn.isClickable()) getMotionBtn.setClickable(false);
```

```
255     isMenuClickable = false;
256     secondTv.setText("0");
257     getMotionBtn.setText("データ処理中");
258
259     LogUtil.log(Log.DEBUG, "Start initialize progress dialog");
260
261     progressDialog = new ProgressDialog(this);
262     progressDialog.setTitle("計算処理中");
263     progressDialog.setMessage("しばらくお待ちください");
264     progressDialog.setIndeterminate(false);
265     progressDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
266     progressDialog.setCancelable(false);
267
268     LogUtil.log(Log.DEBUG, "Finish initialize Progress dialog");
269
270     progressDialog.show();
271
272     // スレッドを作り、開始する(
273     //     runメソッドに飛ぶ)。表面ではプログレスダイアログがくるくる
274     Thread thread = new Thread(this);
275     thread.start();
276 }
277
278 @Override
279 public void run() {
280     LogUtil.log(Log.DEBUG, "Thread running");
281     mManageData.writeFloatThreeArrayData("RegistRawData", "rawAccelo",
282         RegistNameInput.name, accelFloat);
283     mManageData.writeFloatThreeArrayData("RegistRawData", "rawGyro",
284         RegistNameInput.name, gyroFloat);
285
286     resultCalc = calc();
287     resultCorrelation = measureCorrelation();
288
289     progressDialog.dismiss();
290     progressDialog = null;
291     resultHandler.sendMessage(FINISH);
292     LogUtil.log(Log.DEBUG, "Thread finished");
293 }
```

```
293  * データ加工 , 計算処理を行う
294  */
295  private boolean calc () {
296      LogUtil.log(Log.INFO);
297      // データの桁揃え
298      double[][][] accel_double = mFormatter.floatToDoubleFormatter(accelFloat);
299      double[][][] gyro_double = mFormatter.floatToDoubleFormatter(gyroFloat);
300
301      mManageData.writeDoubleThreeArrayData("BeforeAMP", "accel", RegistNameInput
        .name, accel_double);
302      mManageData.writeDoubleThreeArrayData("BeforeAMP", "gyro", RegistNameInput.
        name, gyro_double);
303
304      if (mAmplifier.CheckValueRange(accel_double, checkRangeValue) || mAmplifier
        .CheckValueRange(gyro_double, checkRangeValue)) {
305          accel_double = mAmplifier.Amplify(accel_double, ampValue);
306          gyro_double = mAmplifier.Amplify(gyro_double, ampValue);
307      }
308
309      mManageData.writeDoubleThreeArrayData("AfterAMP", "accel", RegistNameInput.
        name, accel_double);
310      mManageData.writeDoubleThreeArrayData("AfterAMP", "gyro", RegistNameInput.
        name, gyro_double);
311
312
313      // フーリエ変換によるローパスフィルタ
314      accel_double = mFourier.LowpassFilter(accel_double, "accel");
315      gyro_double = mFourier.LowpassFilter(gyro_double, "gyro");
316
317      mManageData.writeDoubleThreeArrayData("AfterLowpass", "accel",
        RegistNameInput.name, accel_double);
318      mManageData.writeDoubleThreeArrayData("AfterLowpass", "gyro",
        RegistNameInput.name, gyro_double);
319
320
321      LogUtil.log(Log.DEBUG, "Finish fourier");
322
323      distance = mCalc.accelToDistance(accel_double, 0.03);
324      angle = mCalc.gyroToAngle(gyro_double, 0.03);
325
326      distance = mFormatter.doubleToDoubleFormatter(distance);
```

```
327     angle = mFormatter.doubleToDoubleFormatter(angle);
328
329     mManageData.writeDoubleThreeArrayData("convertData", "distance",
330         RegistNameInput.name, distance);
331
332     mManageData.writeDoubleThreeArrayData("convertData", "angle",
333         RegistNameInput.name, angle);
334
335     LogUtil.log(Log.DEBUG, "After write data");
336
337     // measureCorrelation用の平均値データを作成
338     for (int i = 0; i < 3; i++) {
339         for (int j = 0; j < 100; j++) {
340             averageDistance[i][j] = (distance[0][i][j] + distance[1][i][j] +
341                 distance[2][i][j]) / 3;
342             averageAngle[i][j] = (angle[0][i][j] + angle[1][i][j] + angle[2][i][j])
343                 / 3;
344         }
345     }
346
347     //region 同一のモーションであるかの確認をし、必要に応じてズレ修正を行う
348     Enum.MEASURE measure = mCorrelation.measureCorrelation(distance, angle,
349         averageDistance, averageAngle);
350
351     LogUtil.log(Log.DEBUG, "After measure correlation");
352     LogUtil.log(Log.DEBUG, "measure = " + String.valueOf(measure));
353
354     if (Enum.MEASURE.BAD == measure) {
355         // 相関係数が0.4以下
356         return false;
357     }
358     else if (Enum.MEASURE.INCORRECT == measure) {
359         LogUtil.log(Log.ERROR, "Deviation");
360         // 相関係数が0.4よりも高く、0.6以下の場合
361         // ズレ修正を行う
362         int time = 0;
363         Enum.MODE mode = Enum.MODE.MAX;
364         Enum.TARGET target = Enum.TARGET.DISTANCE;
365
366         double[][][] originalDistance = distance;
367         double[][][] originalAngle = angle;
```

```
363
364 //ズレ修正は基準値を最大値，最小値，中央値の順に置き，さらに距離，角度の
    順にベースを置く．
365 while (true) {
366     switch (time) {
367         case 0:
368             mode = Enum.MODE.MAX;
369             target = Enum.TARGET.DISTANCE;
370             break;
371         case 1:
372             mode = Enum.MODE.MAX;
373             target = Enum.TARGET.ANGLE;
374             break;
375         case 2:
376             mode = Enum.MODE.MIN;
377             target = Enum.TARGET.DISTANCE;
378             break;
379         case 3:
380             mode = Enum.MODE.MIN;
381             target = Enum.TARGET.ANGLE;
382             break;
383         case 4:
384             mode = Enum.MODE.MEDIAN;
385             target = Enum.TARGET.DISTANCE;
386             break;
387         case 5:
388             mode = Enum.MODE.MEDIAN;
389             target = Enum.TARGET.ANGLE;
390             break;
391     }
392
393     double[][][][] deviatedValue = mCorrectDeviation.correctDeviation(
        originalDistance, originalAngle, mode, target);
394
395     for (int i = 0; i < 3; i++) {
396         for (int j = 0; j < 3; j++) {
397             for (int k = 0; k < 100; k++) {
398                 distance[i][j][k] = deviatedValue[0][i][j][k];
399                 angle[i][j][k] = deviatedValue[1][i][j][k];
400             }
401         }
```



```
402     }
403
404     for (int i = 0; i < 3; i++) {
405         for (int j = 0; j < 100; j++) {
406             averageDistance[i][j] = (distance[0][i][j] + distance[1][i][j] +
407                                     distance[2][i][j]) / 3;
408             averageAngle[i][j] = (angle[0][i][j] + angle[1][i][j] + angle[2][i][j]) / 3;
409         }
410     }
411
412     Enum.MEASURE tmp = mCorrelation.measureCorrelation(distance, angle,
413                                                         averageDistance, averageAngle);
414
415     LogUtil.log(Log.ERROR, "MEASURE: " + String.valueOf(tmp));
416
417     mManageData.writeDoubleThreeArrayData("deviatedData" + String.valueOf(
418         mode), "distance", RegistNameInput.name, distance);
419     mManageData.writeDoubleThreeArrayData("deviatedData" + String.valueOf(
420         mode), "angle", RegistNameInput.name, angle);
421
422     if (tmp == Enum.MEASURE.PERFECT || tmp == Enum.MEASURE.CORRECT) {
423         break;
424     }
425     else if (time == 2) {
426         // 相関係数が低いまま，アラートなどを出す？
427         distance = originalDistance;
428         angle = originalAngle;
429         break;
430     }
431
432     time++;
433 }
434
435 else if (Enum.MEASURE.PERFECT == measure || Enum.MEASURE.CORRECT == measure) {
436     // PERFECTなら，何もしない
437 }
438
439 else {
440     // なにかがおかしい
```

```
437     return false;
438 }
439 //endregion
440
441 mManageData.writeDoubleThreeArrayData("AfterCalcData", "afterFormatDistance",
    ", RegistNameInput.name, distance);
442 mManageData.writeDoubleThreeArrayData("AfterCalcData", "afterFormatAngle",
    RegistNameInput.name, angle);
443
444 // ブレ修正後の平均値データを出す
445 for (int i = 0; i < 3; i++) {
446     for (int j = 0; j < 100; j++) {
447         averageDistance[i][j] = (distance[0][i][j] + distance[1][i][j] +
            distance[2][i][j]) / 3;
448         averageAngle[i][j] = (angle[0][i][j] + angle[1][i][j] + angle[2][i][j])
            / 3;
449     }
450 }
451
452 LogUtil.log(Log.DEBUG, "return");
453 return true;
454 }
455
456 /**
457  * 相関係数を導出し，ユーザが入力した3回のモーションの類似性を確認する
458  */
459 private boolean measureCorrelation () {
460     LogUtil.log(Log.INFO);
461     Enum.MEASURE measure = mCorrelation.measureCorrelation(distance, angle,
        averageDistance, averageAngle);
462
463     LogUtil.log(Log.DEBUG, "measure = " + measure);
464
465     return measure == Enum.MEASURE.CORRECT || measure == Enum.MEASURE.PERFECT;
466 }
467
468 /**
469  * 計算，モーション照合処理終了後に呼ばれるハンドラ
470  * 同一のモーションであると確認されたら登録を行い，そうでなければ取り直しの処
    理を行う
```

```
472     */
473     private Handler resultHandler = new Handler() {
474         public void handleMessage (Message msg) {
475             if (msg.what == FINISH) {
476                 if (!resultCalc || !resultCorrelation) {
477                     // もう一度モーションを取り直す処理
478                     // ボタンのstatusをenableにして押せるようにする
479                     AlertDialog.Builder alert = new AlertDialog.Builder(RegistMotion.this);
480                     alert.setOnKeyListener(new DialogInterface.OnKeyListener() {
481                         @Override
482                         public boolean onKey (DialogInterface dialog, int keyCode, KeyEvent
483                             event) {
484                             return keyCode == KeyEvent.KEYCODE_BACK;
485                         }
486                     });
487                     alert.setCancelable(false);
488                     alert.setTitle("登録失敗");
489                     alert.setMessage("登録に失敗しました。やり直して下さい");
490                     alert.setNeutralButton("OK", new DialogInterface.OnClickListener() {
491                         @Override
492                         public void onClick (DialogInterface dialog, int which) {
493                             resetValue();
494                         }
495                     });
496                     alert.show();
497                 }
498             }
499         }
500     }
501     else {
502         // 3回のモーションの平均値をファイルに書き出す
503         mManageData.writeRegisteredData(RegistNameInput.name, averageDistance,
504             averageAngle, ampValue, RegistMotion.this);
505     }
506     AlertDialog.Builder alert = new AlertDialog.Builder(RegistMotion.this);
507     alert.setOnKeyListener(new DialogInterface.OnKeyListener() {
508         @Override
```

```
509         public boolean onKey (DialogInterface dialog, int keyCode, KeyEvent
510             event) {
511             return keyCode == KeyEvent.KEYCODE.BACK;
512         }
513     });
514
515     alert.setCancelable(false);
516
517     alert.setTitle("登録完了");
518     alert.setMessage("登録が完了しました。 \n スタート画面に戻ります");
519
520     alert.setNeutralButton("OK", new DialogInterface.OnClickListener() {
521         @Override
522         public void onClick (DialogInterface dialog, int which) {
523             finishRegist();
524         }
525     });
526
527     alert.show();
528 }
529 }
530 };
531
532
533 /**
534  * モーション取得に関する変数群を初期化する
535  */
536 private void resetValue () {
537     getMotionBtn.setClickable(true);
538     // データ取得関係の変数を初期化
539     dataCount = 0;
540     getCount = 0;
541     secondTv.setText("3");
542     getMotionBtn.setText("モーションデータ取得");
543 }
544
545
546 @Override
547 public void onAccuracyChanged (Sensor sensor, int accuracy) {
548     LogUtil.log(Log.INFO);
```



```
587
588 // 閾値調整
589 SeekBar thresholdSeekBar = (SeekBar) seekView.findViewById(R.id.
    threshold);
590 final TextView thresholdSeekText = (TextView) seekView.findViewById(R
    .id.thresholdtext);
591 thresholdSeekText.setText("増幅器にかけるかどうかを判断する閾値を調整できます。
    \n" +
592         "2.5を中心に、値が小さければ登録・認証が難
            しくなり、大きければ易しくなります。 \n"
        +
593         "現在の値は" + checkRangeValue + "です。");
594
595 thresholdSeekBar.setMax(30);
596
597 thresholdSeekBar.setProgress(16);
598 thresholdSeekBar.setOnSeekBarChangeListener(new SeekBar.
    OnSeekBarChangeListener() {
599     @Override
600     public void onProgressChanged (SeekBar seekBar, int progress,
        boolean fromUser) {
601         checkRangeValue = (seekBar.getProgress() + 10) / 10.0;
602         thresholdSeekText.setText("増幅器にかけるかどうかを判断する閾値を調整できま
            す。 \n" +
603             "2.5を中心に、値が小さければ登録・認証
                が難しくなり、大きければ易しくなりま
                    す。 \n" +
604             "現在の値は" + checkRangeValue + "です。");
605     }
606
607     @Override
608     public void onStartTrackingTouch (SeekBar seekBar) {
609     }
610
611     @Override
612     public void onStopTrackingTouch (SeekBar seekBar) {
613         checkRangeValue = (seekBar.getProgress() + 10) / 10.0;
614         thresholdSeekText.setText("増幅器にかけるかどうかを判断する閾値を調整できま
            す。 \n" +
615             "2.5を中心に、値が小さければ登録・認証
                が難しくなり、大きければ易しくなりま
                    す。 \n" +
616             "現在の値は" + checkRangeValue + "です。");
```

```
617     }
618   });
619
620   // 増幅値調整
621   SeekBar ampvalSeekBar = (SeekBar) seekView.findViewById(R.id.ampval);
622   final TextView ampvalText = (TextView) seekView.findViewById(R.id.
       ampvaltext);
623   ampvalText.setText("増幅器にかける場合に、何倍増幅するかを調整できます。標準
       は2倍です。\\n" +
624                       "現在の値は" + ampValue + "です。");
625
626   ampvalSeekBar.setMax(10);
627
628   ampvalSeekBar.setProgress(2);
629   ampvalSeekBar.setOnSeekBarChangeListener(new SeekBar.
       OnSeekBarChangeListener() {
630       @Override
631       public void onProgressChanged (SeekBar seekBar, int progress,
           boolean fromUser) {
632           ampValue = seekBar.getProgress() * 1.0;
633           ampvalText.setText("増幅器にかける場合に、何倍増幅するかを調整できます。標準
           は2倍です。\\n" +
634                               "現在の値は" + ampValue + "です。");
635       }
636
637       @Override
638       public void onStartTrackingTouch (SeekBar seekBar) {
639       }
640
641       @Override
642       public void onStopTrackingTouch (SeekBar seekBar) {
643           ampValue = seekBar.getProgress() * 1.0;
644           ampvalText.setText("増幅器にかける場合に、何倍増幅するかを調整できます。標準
           は2倍です。\\n" +
645                               "現在の値は" + ampValue + "です。");
646       }
647   });
648
649   AlertDialog.Builder dialog = new AlertDialog.Builder(RegistMotion.
       this);
650   dialog.setOnKeyListener(new DialogInterface.OnKeyListener() {
651       @Override
```

```
652         public boolean onKey (DialogInterface dialog1, int keyCode,
653                               KeyEvent event) {
654             return keyCode == KeyEvent.KEYCODE.BACK;
655         }
656     });
657     dialog.setTitle("増幅器設定");
658     dialog.setView(seekView);
659     dialog.setCancelable(false);
660     dialog.setPositiveButton("OK", new DialogInterface.OnClickListener()
661     {
662         @Override
663         public void onClick (DialogInterface dialog1, int which) {
664             dialog.show();
665         }
666     });
667     return true;
668
669     case R.id.reset:
670         if (isMenuClickable) {
671             AlertDialog.Builder alert = new AlertDialog.Builder(RegistMotion.this);
672             alert.setOnKeyListener(new DialogInterface.OnKeyListener() {
673                 @Override
674                 public boolean onKey (DialogInterface dialog, int keyCode, KeyEvent
675                                     event) {
676                     return keyCode == KeyEvent.KEYCODE.BACK;
677                 }
678             });
679             alert.setCancelable(false);
680             alert.setTitle("データ取得リセット");
681             alert.setMessage("本当にデータ取得をやり直しますか？ ");
682             alert.setNegativeButton("NO", new DialogInterface.OnClickListener() {
683                 @Override
684                 public void onClick (DialogInterface dialog, int which) {
685                     }
686             });
687         }
688     }
```



```
689         alert.setPositiveButton("YES", new DialogInterface.OnClickListener()
690             {
691                 @Override
692                 public void onClick (DialogInterface dialog, int which) {
693                     RegistMotion.this.resetValue();
694                 }
695             });
696         alert.show();
697     }
698     return true;
699 }
700 return false;
701 }
702
703
704 /**
705  * スタート画面に移動するメソッド
706  */
707 private void finishRegist () {
708     LogUtil.log(Log.INFO);
709     Intent intent = new Intent();
710
711
712     intent.setClassName("com.example.motionauth", "com.example.motionauth.Start
713         ");
714
715     intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.
716         FLAG_ACTIVITY_NEW_TASK);
717
718     startActivityForResult(intent, 0);
719     finish();
720 }
```

C.4 src/com/example/motionauth/Authentication/AuthNameInput.java

```
1 package com.example.motionauth.Authentication;
2
3 import android.app.Activity;
4 import android.content.Context;
```

```
5 import android.content.Intent;
6 import android.content.SharedPreferences;
7 import android.os.Bundle;
8 import android.text.Editable;
9 import android.text.TextWatcher;
10 import android.util.Log;
11 import android.view.KeyEvent;
12 import android.view.View;
13 import android.view.Window;
14 import android.view.inputmethod.InputMethodManager;
15 import android.widget.Button;
16 import android.widget.EditText;
17 import android.widget.Toast;
18 import com.example.motionauth.R;
19 import com.example.motionauth.Utility.LogUtil;
20
21
22 /**
23  * 認証するユーザ名を入力させる
24  *
25  * @author Kensuke Kousaka
26  */
27 public class AuthNameInput extends Activity {
28     // ユーザが入力した文字列（名前）を格納する
29     public static String name;
30
31     private Context current;
32
33
34     @Override
35     protected void onCreate(Bundle savedInstanceState) {
36         super.onCreate(savedInstanceState);
37
38         LogUtil.log(Log.INFO);
39
40         // タイトルバーの非表示
41         requestWindowFeature(Window.FEATURE_NO_TITLE);
42         setContentView(R.layout.activity_auth_name_input);
43         current = this;
44
45         name = "";
```

```
46
47     nameInput();
48 }
49
50
51 /**
52  * ユーザ名を入力させる
53  */
54 private void nameInput () {
55     LogUtil.log(Log.INFO);
56     final EditText nameInput = (EditText) findViewById(R.id.nameInputEditText);
57
58     nameInput.addTextChangedListener(new TextWatcher() {
59         // 変更前
60         public void beforeTextChanged (CharSequence s, int start, int count, int
61             after) {
62
63         }
64
65         // 変更直前
66         public void onTextChanged (CharSequence s, int start, int before, int
67             count) {
68
69         }
70
71         // 変更後
72         public void afterTextChanged (Editable s) {
73             if (nameInput.getText() != null) name = nameInput.getText().toString().
74                 trim();
75         }
76     });
77
78     // ソフトウェアキーボードの
79     // Enterキーを押した際に、ソフトウェアキーボードを閉じるようにする
80     nameInput.setOnKeyListener(new View.OnKeyListener() {
81         @Override
82         public boolean onKey (View v, int keyCode, KeyEvent event) {
83             if (event.getAction() == KeyEvent.ACTION_DOWN && keyCode == KeyEvent.
84                 KEYCODE_ENTER) {
85                 LogUtil.log(Log.DEBUG, "Push enter key");
86                 InputMethodManager inputMethodManager = (InputMethodManager)
87                     AuthNameInput.this.getSystemService(Context.INPUT_METHOD_SERVICE);
88                 inputMethodManager.hideSoftInputFromWindow(v.getWindowToken(), 0);
89             }
90         }
91     });
92 }
```

```
81
82         return true;
83     }
84     return false;
85 }
86 });
87
88 // OKボタンを押した時に、次のアクティビティに移動
89 final Button ok = (Button) findViewById(R.id.okButton);
90
91 ok.setOnClickListener(new View.OnClickListener() {
92     @Override
93     public void onClick (View v) {
94         LogUtil.log(Log.DEBUG, "Click ok button");
95
96         // 指定したユーザが存在するかどうかを確認する
97         if (AuthNameInput.this.checkUserExists()) {
98             LogUtil.log(Log.DEBUG, "User is existed");
99             AuthNameInput.this.moveActivity("com.example.motionauth", "com.
               example.motionauth.Authentication.AuthMotion", true);
100        }
101        else {
102            LogUtil.log(Log.DEBUG, "User is not existed");
103            Toast.makeText(current, "ユーザが登録されていませ
               ん", Toast.LENGTHLONG).show();
104        }
105    }
106 });
107 }
108
109
110 /**
111  * 入力したユーザが以前に登録したことのあるユーザかどうかを確認 データがない
  　　のに認証はできない
112  *
113  * @return 登録したことがあるユーザであればtrue, 登録したことがなければfalse
114  */
115 private boolean checkUserExists () {
116     LogUtil.log(Log.INFO);
117
118     Context mContext = AuthNameInput.this.getApplicationContext();
```

```
119     SharedPreferences preferences = mContext.getSharedPreferences("UserList",
120                               Context.MODE_PRIVATE);
121     return preferences.contains(name);
122 }
123
124
125 /**
126  * アクティビティを移動する
127  *
128  * @param pkgName 移動先のパッケージ名
129  * @param actName 移動先のアクティビティ名
130  * @param flg      戻るキーを押した際にこのアクティビティを表示させるかどうか
131  */
132 private void moveActivity (String pkgName, String actName, boolean flg) {
133     LogUtil.log(Log.INFO);
134     Intent intent = new Intent();
135
136     intent.setClassName(pkgName, actName);
137
138     if (flg) intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.
139                               FLAG_ACTIVITY_NEW_TASK);
140
141     startActivityForResult(intent, 0);
142 }
```

C.5 src/com/example/motionauth/Authentication/AuthMotion.java

```
1 package com.example.motionauth.Authentication;
2
3 import android.app.Activity;
4 import android.app.AlertDialog;
5 import android.app.ProgressDialog;
6 import android.content.Context;
7 import android.content.DialogInterface;
8 import android.content.Intent;
9 import android.content.SharedPreferences;
10 import android.hardware.Sensor;
11 import android.hardware.SensorEvent;
12 import android.hardware.SensorEventListener;
```

```
13 import android.hardware.SensorManager;
14 import android.os.Bundle;
15 import android.os.Handler;
16 import android.os.Message;
17 import android.os.Vibrator;
18 import android.util.Log;
19 import android.view.View;
20 import android.widget.Button;
21 import android.widget.TextView;
22 import com.example.motionauth.Lowpass.Fourier;
23 import com.example.motionauth.Processing.Amplifier;
24 import com.example.motionauth.Processing.Calc;
25 import com.example.motionauth.Processing.Correlation;
26 import com.example.motionauth.Processing.Formatter;
27 import com.example.motionauth.R;
28 import com.example.motionauth.Utility.Enum;
29 import com.example.motionauth.Utility.LogUtil;
30 import com.example.motionauth.Utility.ManageData;
31
32 import java.util.ArrayList;
33
34 /**
35  * ユーザ認証を行う
36  *
37  * @author Kensuke Kousaka
38  */
39 public class AuthMotion extends Activity implements SensorEventListener,
    Runnable {
40     private static final int VIBRATOR.SHORT = 25;
41     private static final int VIBRATOR.NORMAL = 50;
42     private static final int VIBRATOR.LONG = 100;
43
44     private static final int PREPARATION = 1;
45     private static final int GET_MOTION = 2;
46
47     private static final int PREPARATION.INTERVAL = 1000;
48     private static final int GET_MOTION.INTERVAL = 30;
49
50     private static final int FINISH = 5;
51
52     private SensorManager mSensorManager;
```

```
53 private Sensor      mAccelerometerSensor;
54 private Sensor      mGyroscopeSensor;
55
56 private Vibrator mVibrator;
57
58 private TextView secondTv;
59 private TextView countSecondTv;
60 private Button    getMotionBtn;
61
62 private Fourier    mFourier    = new Fourier();
63 private Formatter  mFormatter  = new Formatter();
64 private Calc       mCalc       = new Calc();
65 private Correlation mCorrelation = new Correlation();
66 private Amplifier  mAmplifier  = new Amplifier();
67
68 private int dataCount    = 0;
69 private int prepareCount = 0;
70
71 private boolean isGetMotionBtnClickable = true;
72
73 private double ampValue = 0.0;
74
75 // モーションの生データ
76 private float[] vAccel;
77 private float[] vGyro;
78
79 private float[][] accelFloat = new float[3][100];
80 private float[][] gyroFloat  = new float[3][100];
81
82 private double[][] distance = new double[3][100];
83 private double[][] angle    = new double[3][100];
84
85 private double[][] registered_ave_distance = new double[3][100];
86 private double[][] registered_ave_angle    = new double[3][100];
87
88 private boolean resultCorrelation = false;
89
90 private ProgressDialog progressDialog;
91
92
93 @Override
```

```
94  protected void onCreate (Bundle savedInstanceState) {
95      super.onCreate(savedInstanceState);
96
97      LogUtil.log(Log.INFO);
98
99      setContentView(R.layout.activity_auth_motion);
100
101      authMotion();
102  }
103
104  /**
105   * 認証画面にイベントリスナ等を設定する
106   */
107  private void authMotion () {
108      LogUtil.log(Log.INFO);
109
110      // センササービス, 各種センサを取得する
111      mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
112      mAccelerometerSensor = mSensorManager.getDefaultSensor(Sensor.
          TYPE_ACCELEROMETER);
113      mGyroscopeSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
114
115      mVibrator = (Vibrator) getSystemService(VIBRATOR.SERVICE);
116
117      TextView nameTv = (TextView) findViewById(R.id.textView1);
118      secondTv = (TextView) findViewById(R.id.secondTextView);
119      countSecondTv = (TextView) findViewById(R.id.textView4);
120      getMotionBtn = (Button) findViewById(R.id.button1);
121
122      nameTv.setText(AuthNameInput.name + "さん読んでね! ");
123
124      getMotionBtn.setOnClickListener(new View.OnClickListener() {
125          @Override
126          public void onClick (View v) {
127              LogUtil.log(Log.DEBUG, "Click get motion button");
128              if (isGetMotionBtnClickable) {
129                  isGetMotionBtnClickable = false;
130
131                  // ボタンをクリックできないようにする
132                  v.setClickable(false);
133
```



```
134         getMotionBtn.setText("インターバル中");
135         countSecondTv.setText("秒");
136
137         // timeHandler呼び出し
138         timeHandler.sendMessage(PREPARATION);
139     }
140 }
141 });
142 }
143
144
145 /**
146  * 一定時間ごとにモーションデータを取得するハンドラ
147  * 計算処理や相関係数計算関数もここから呼び出す
148  */
149 Handler timeHandler = new Handler() {
150     @Override
151     public void dispatchMessage (Message msg) {
152         if (msg.what == PREPARATION && !isGetMotionBtnClickable) {
153             switch (prepareCount) {
154                 case 0:
155                     secondTv.setText("3");
156                     mVibrator.vibrate(VIBRATOR.SHORT);
157
158                     // 第二引数で指定したミリ秒分遅延させてから、第一引数のメッセージを
159                     // 添えてtimeHandlerを呼び出す
160                     timeHandler.sendMessageDelayed(PREPARATION,
161                                                     PREPARATION_INTERVAL);
162                     break;
163                 case 1:
164                     secondTv.setText("2");
165                     mVibrator.vibrate(VIBRATOR.SHORT);
166                     timeHandler.sendMessageDelayed(PREPARATION,
167                                                     PREPARATION_INTERVAL);
168                     break;
169                 case 2:
170                     secondTv.setText("1");
171                     mVibrator.vibrate(VIBRATOR.SHORT);
172                     timeHandler.sendMessageDelayed(PREPARATION,
173                                                     PREPARATION_INTERVAL);
174                     break;
```

```
171         case 3:
172             secondTv.setText("START");
173             mVibrator.vibrate(VIBRATOR.LONG);
174
175             // GET_MOTIONメッセージを添えて, timeHandlerを呼び出す
176             timeHandler.sendMessage(GET_MOTION);
177             getMotionBtn.setText("取得中");
178             break;
179     }
180
181     prepareCount++;
182 }
183 else if (msg.what == GET_MOTION && !isGetMotionBtnClickable) {
184     if (dataCount < 100) {
185         // 取得した値を, 0.03秒ごとに配列に入れる
186         for (int i = 0; i < 3; i++) {
187             accelFloat[i][dataCount] = vAccel[i];
188             gyroFloat[i][dataCount] = vGyro[i];
189         }
190
191         dataCount++;
192
193         switch (dataCount) {
194             case 1:
195                 secondTv.setText("3");
196                 mVibrator.vibrate(VIBRATOR.NORMAL);
197                 break;
198             case 33:
199                 secondTv.setText("2");
200                 mVibrator.vibrate(VIBRATOR.NORMAL);
201                 break;
202             case 66:
203                 secondTv.setText("1");
204                 mVibrator.vibrate(VIBRATOR.NORMAL);
205                 break;
206         }
207
208         timeHandler.sendMessageDelayed(GET_MOTION, GET_MOTION_INTERVAL);
209     }
210     else if (dataCount >= 100) {
211         mVibrator.vibrate(VIBRATOR.LONG);
```

```
212         finishGetMotion();
213     }
214 }
215 else {
216     super.dispatchMessage(msg);
217 }
218 }
219 };
220
221
222 @Override
223 public void onSensorChanged (SensorEvent event) {
224     if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) vAccel = event.
        values.clone();
225     if (event.sensor.getType() == Sensor.TYPE_GYROSCOPE) vGyro = event.values.
        clone();
226 }
227
228 private void finishGetMotion () {
229     getMotionBtn.setText("認証処理中");
230
231     prepareCount = 0;
232
233     LogUtil.log(Log.DEBUG, "Start initialize ProgressDialog");
234
235     progressDialog = new ProgressDialog(this);
236     progressDialog.setTitle("計算処理中");
237     progressDialog.setMessage("しばらくお待ちください");
238     progressDialog.setIndeterminate(false);
239     progressDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
240     progressDialog.setCancelable(false);
241
242     LogUtil.log(Log.DEBUG, "Finish initialize ProgressDialog");
243
244     progressDialog.show();
245
246     // スレッドを作り、開始する(
        runメソッドに飛ぶ)。表面ではプログレスダイアログがくるくる
247     Thread thread = new Thread(this);
248     thread.start();
249 }
```

```
250
251 @Override
252 public void run () {
253     LogUtil.log(Log.DEBUG, "Thread start");
254
255     readRegisteredData();
256     calc();
257     resultCorrelation = measureCorrelation();
258
259     progressDialog.dismiss();
260     progressDialog = null;
261     resultHandler.sendMessage(FINISH);
262
263     LogUtil.log(Log.DEBUG, "Thread finish");
264 }
265
266 /**
267  * RegistMotionにて登録したモーションの平均値データを読み込む
268  */
269 private void readRegisteredData () {
270     LogUtil.log(Log.INFO);
271
272     ManageData mManageData = new ManageData();
273     ArrayList<double[][]> readDataList = mManageData.readRegisteredData(
274         AuthMotion.this, AuthNameInput.name);
275
276     registered_ave_distance = readDataList.get(0);
277     registered_ave_angle = readDataList.get(1);
278
279     SharedPreferences preferences = AuthMotion.this.getApplicationContext().
280         getSharedPreferences("MotionAuth", Context.MODE_PRIVATE);
281     String registeredAmplify = preferences.getString(AuthNameInput.name + "
282         amplify", "");
283
284     if ("".equals(registeredAmplify)) throw new RuntimeException();
285
286     ampValue = Double.valueOf(registeredAmplify);
287 }
288
289 /**
290  * データ加工・計算処理を行う
```

```
288     */
289     private void calc () {
290         LogUtil.log(Log.INFO);
291         // 原データの桁揃え
292         double[][] accel = mFormatter.floatToDoubleFormatter(accelFloat);
293         double[][] gyro = mFormatter.floatToDoubleFormatter(gyroFloat);
294
295         // if (isAmplify) {
296         LogUtil.log(Log.DEBUG, "Amplify on");
297         accel = mAmplifier.Amplify(accel, ampValue);
298         gyro = mAmplifier.Amplify(gyro, ampValue);
299         // }
300
301         // フーリエ変換を用いたローパス処理
302         accel = mFourier.LowpassFilter(accel, "accel");
303         gyro = mFourier.LowpassFilter(gyro, "gyro");
304
305         distance = mCalc.accelToDistance(accel, 0.03);
306         angle = mCalc.gyroToAngle(gyro, 0.03);
307
308         distance = mFormatter.doubleToDoubleFormatter(distance);
309         angle = mFormatter.doubleToDoubleFormatter(angle);
310     }
311
312
313     /**
314     * 認証処理終了後に呼び出されるハンドラ
315     * 認証に成功すればスタート画面に戻り、そうでなければ認証やり直しの処理を行う
316     */
317     private Handler resultHandler = new Handler() {
318         public void handleMessage (Message msg) {
319             if (msg.what == FINISH) {
320                 if (!resultCorrelation) {
321                     LogUtil.log(Log.INFO, "False authentication");
322                     AlertDialog.Builder alert = new AlertDialog.Builder(AuthMotion.this);
323                     alert.setTitle("認証失敗です");
324                     alert.setMessage("認証に失敗しました");
325                     alert.setCancelable(false);
326                     alert.setNeutralButton("OK", new DialogInterface.OnClickListener() {
327                         @Override
328                         public void onClick (DialogInterface dialog, int which) {
```

```
329         isGetMotionBtnClickable = true;
330         getMotionBtn.setClickable(true);
331         // データ取得関係の変数を初期化
332         dataCount = 0;
333         secondTv.setText("3");
334         getMotionBtn.setText("モーションデータ取得");
335     }
336 });
337 alert.show();
338 }
339 else {
340     LogUtil.log(Log.INFO, "Success authentication");
341     AlertDialog.Builder alert = new AlertDialog.Builder(AuthMotion.this);
342     alert.setTitle("認証成功");
343     alert.setMessage("認証に成功しました.\nスタート画面に戻ります.");
344     alert.setCancelable(false);
345     alert.setNeutralButton("OK", new DialogInterface.OnClickListener() {
346         @Override
347         public void onClick (DialogInterface dialog, int which) {
348             moveActivity("com.example.motionauth", "com.example.motionauth.
349                 Start", true);
350         }
351     });
352     alert.show();
353 }
354 }
355 };
356
357
358 private boolean measureCorrelation () {
359     LogUtil.log(Log.INFO);
360     Enum.MEASURE measure = mCorrelation.measureCorrelation(distance, angle,
361         registred_ave_distance, registred_ave_angle);
362
363     LogUtil.log(Log.DEBUG, "measure: " + measure);
364
365     return measure == Enum.MEASURE.CORRECT;
366 }
367 @Override
```

```
368 public void onAccuracyChanged (Sensor sensor, int accuracy) {
369     LogUtil.log(Log.INFO);
370 }
371
372
373 @Override
374 protected void onResume () {
375     super.onResume();
376     LogUtil.log(Log.INFO);
377
378     mSensorManager.registerListener(this, mAccelerometerSensor, SensorManager.
        SENSOR_DELAY_GAME);
379     mSensorManager.registerListener(this, mGyroscopeSensor, SensorManager.
        SENSOR_DELAY_GAME);
380 }
381
382
383 @Override
384 protected void onPause () {
385     super.onPause();
386     LogUtil.log(Log.INFO);
387
388     mSensorManager.unregisterListener(this);
389 }
390
391
392 /**
393  * アクティビティを移動する
394  *
395  * @param pkgName 移動先のパッケージ名
396  * @param actName 移動先のアクティビティ名
397  * @param flg      戻るキーを押した際にこのアクティビティを表示させるかどうか
398  */
399 private void moveActivity (String pkgName, String actName, boolean flg) {
400     LogUtil.log(Log.INFO);
401     Intent intent = new Intent();
402
403     intent.setClassName(pkgName, actName);
404
405     if (flg) intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.
        FLAG_ACTIVITY_NEW_TASK);
```

```
406
407     startActivityForResult(intent, 0);
408     finish();
409 }
410 }
```

C.6 src/com/example/motionauth/ViewDataList/RegistrantList.java

```
1  package com.example.motionauth.ViewDataList;
2
3  import android.app.Activity;
4  import android.app.AlertDialog;
5  import android.content.Context;
6  import android.content.DialogInterface;
7  import android.content.Intent;
8  import android.content.SharedPreferences;
9  import android.os.Bundle;
10 import android.util.Log;
11 import android.view.View;
12 import android.view.Window;
13 import android.widget.AdapterView;
14 import android.widget.ArrayAdapter;
15 import android.widget.ListView;
16 import com.example.motionauth.R;
17 import com.example.motionauth.Utility.LogUtil;
18
19 import java.util.ArrayList;
20 import java.util.Map;
21
22
23 /**
24  * 登録されているユーザ名を一覧表示する .
25  * ユーザ名が選択されたら , そのユーザのデータを
26  *   ViewRegistedDataアクティビティにて表示する
27  *
28  * @author Kensuke Kousaka
29  */
30 public class RegistrantList extends Activity {
31     String item;
32 }
```



```
33  @Override
34  protected void onCreate (Bundle savedInstanceState) {
35      super.onCreate(savedInstanceState);
36
37      LogUtil.log(Log.INFO);
38
39      // タイトルバーの非表示
40      requestWindowFeature(Window.FEATURE_NO_TITLE);
41      setContentView(R.layout.activity_registrant_list);
42
43      registrantList();
44  }
45
46
47  /**
48   * 登録されているユーザ名のリストを表示する
49   * ユーザ名が選択されたら、そのユーザ名を ViewRegisteredData に送る
50   */
51  private void registrantList () {
52      LogUtil.log(Log.INFO);
53
54      // 登録されているユーザ名のリストを作成する
55      ArrayList<String> userList = getRegistrantName();
56
57      final ListView lv = (ListView) findViewById(R.id.listView1);
58
59      ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.
        simple_list_item_1);
60
61      try {
62          // アイテム追加
63          for (String s : userList) adapter.add(s);
64      }
65      catch (NullPointerException e) {
66          AlertDialog.Builder alert = new AlertDialog.Builder(RegistrantList.this);
67          alert.setTitle("エラー");
68          alert.setMessage("登録されていないユーザです。 \n スタート画面に戻ります。");
69          alert.setCancelable(false);
70          alert.setNeutralButton("OK", new DialogInterface.OnClickListener() {
71              @Override
72              public void onClick (DialogInterface dialog, int which) {
```

```
73         RegistrantList.this.moveActivity("com.example.motionauth", "com.
           example.motionauth.Start", true);
74     }
75     });
76     alert.show();
77     finish();
78 }
79
80 // リストビューにアダプタを設定
81 lv.setAdapter(adapter);
82
83 // リストビューのアイテムがクリックされた時
84 lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
85     @Override
86     public void onItemClick (AdapterView<?> parent, View v, int position,
           long id) {
87         LogUtil.log(Log.DEBUG, "Click item");
88
89         // クリックされたアイテムを取得
90         item = lv.getItemAtPosition(position).toString();
91
92         // itemを次のアクティビティに送る
93         RegistrantList.this.moveActivity("com.example.motionauth", "com.example
           .motionauth.ViewDataList.ViewRegisteredData", true);
94     }
95 });
96 }
97
98
99 /**
100  * 指定されたディレクトリ以下のファイルリストを作成する
101  *
102  * @return 作成されたString配列型のリスト
103  */
104 private ArrayList<String> getRegistrantName () {
105     LogUtil.log(Log.INFO);
106
107     Context mContext = RegistrantList.this.getApplicationContext();
108     SharedPreferences preferences = mContext.getSharedPreferences("UserList",
           Context.MODE_PRIVATE);
109 }
```

```
110     ArrayList<String> keyList = new ArrayList<>();
111
112     Map<String, ?> allEntries = preferences.getAll();
113     for (Map.Entry<String, ?> entry : allEntries.entrySet()) keyList.add(entry.
        getKey());
114
115     return keyList;
116 }
117
118
119 /**
120  * アクティビティを移動する
121  *
122  * @param pkgName 移動先のパッケージ名
123  * @param actName 移動先のアクティビティ名
124  * @param flg      戻るキーを押した際にこのアクティビティを表示させるかどうか
125  */
126 private void moveActivity (String pkgName, String actName, boolean flg) {
127     LogUtil.log(Log.INFO);
128
129     Intent intent = new Intent();
130
131     intent.setClassName(pkgName, actName);
132
133     if (actName.equals("com.example.motionauth.ViewDataList.ViewRegisteredData"))
134         intent.putExtra("item", item);
135
136     if (flg) intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.
137         FLAG_ACTIVITY_NEW_TASK);
138
139     startActivityForResult(intent, 0);
140 }
```

C.7 src/com/example/motionauth/ViewDataList/ViewRegisteredData.java

```
1 package com.example.motionauth.ViewDataList;
2
3 import android.app.ActionBar;
4 import android.app.Activity;
5 import android.content.Context;
```

```
6  import android.content.Intent;
7  import android.content.SharedPreferences;
8  import android.os.Build;
9  import android.os.Bundle;
10 import android.util.Log;
11 import android.view.MenuItem;
12 import android.widget.AdapterView;
13 import android.widget.ListView;
14 import com.example.motionauth.R;
15 import com.example.motionauth.Utility.LogUtil;
16 import com.example.motionauth.Utility.ManageData;
17
18 import java.util.ArrayList;
19
20
21 /**
22  * RegistrantListより渡されたユーザ名を元に、そのユーザのデータを表示する
23  *
24  * @author Kensuke Kousaka
25  */
26 public class ViewRegisteredData extends Activity {
27     String item = null;
28     int flgCount;
29
30
31     @Override
32     protected void onCreate(Bundle savedInstanceState) {
33         super.onCreate(savedInstanceState);
34
35         LogUtil.log(Log.INFO);
36
37         setContentView(R.layout.activity_view_registered_data);
38
39         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.ICE_CREAM_SANDWICH) {
40             ActionBar actionBar = getActionBar();
41             if (actionBar != null) actionBar.setHomeButtonEnabled(true);
42         }
43         flgCount = 0;
44
45         viewRegisteredData();
46     }
```

```
47
48
49 /**
50  * ユーザのデータをリスト表示する
51  */
52 private void viewRegisteredData () {
53     LogUtil.log(Log.INFO);
54
55     // RegistrantListから渡されたユーザ名を受け取る
56     Intent intent = getIntent();
57     item = intent.getStringExtra("item");
58
59     ListView lv = (ListView) findViewById(R.id.listView1);
60
61     ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.
        simple_list_item_1);
62
63     ArrayList<String> dataList = readData();
64
65     // アイテム追加
66     for (String i : dataList) adapter.add(i);
67
68     // リストビューにアダプタを設定
69     lv.setAdapter(adapter);
70 }
71
72
73 /**
74  * データを読み取る
75  *
76  * @return 取得したデータ
77  */
78 private ArrayList<String> readData () {
79     LogUtil.log(Log.INFO);
80     ArrayList<String> dataList = new ArrayList<>();
81
82     ManageData mManageData = new ManageData();
83     ArrayList<double[][]> readData = mManageData.readRegisteredData(
        ViewRegisteredData.this, item);
84     double[][] readDistance = readData.get(0);
85     double[][] readAngle = readData.get(1);
```

```
86
87 String[][] registeredDistance = new String[3][100], registeredAngle = new
    String[3][100];
88 for (int i = 0; i < readDistance.length; i++) {
89     for (int j = 0; j < readDistance[i].length; j++) {
90         registeredDistance[i][j] = String.valueOf(readDistance[i][j]);
91         registeredAngle[i][j] = String.valueOf(readAngle[i][j]);
92     }
93 }
94
95 Context mContext = ViewRegisteredData.this.getApplicationContext();
96 SharedPreferences preferences = mContext.getSharedPreferences("MotionAuth",
    Context.MODE_PRIVATE);
97
98 String ampValue = preferences.getString(item + "amplify", "");
99
100 if (!"".equals(ampValue)) throw new RuntimeException();
101
102 String index = "";
103
104 for (int i = 0; i < registeredDistance.length; i++) {
105     switch (i) {
106         case 0:
107             index = "DistanceX";
108             break;
109         case 1:
110             index = "DistanceY";
111             break;
112         case 2:
113             index = "DistanceZ";
114             break;
115     }
116     for (int j = 0; j < registeredDistance[i].length; j++) {
117         dataList.add(index + " : " + registeredDistance[i][j] + " : " + ampValue
            );
118     }
119 }
120
121 for (int i = 0; i < registeredAngle.length; i++) {
122     switch (i) {
123         case 0:
```

```
124         index = "AngleX";
125         break;
126     case 1:
127         index = "AngleY";
128         break;
129     case 2:
130         index = "AngleZ";
131         break;
132     }
133     for (int j = 0; j < registeredAngle[i].length; j++) {
134         dataList.add(index + " : " + registeredAngle[i][j] + " : " + ampValue);
135     }
136 }
137
138 return dataList;
139 }
140
141
142 @Override
143 public boolean onOptionsItemSelected (MenuItem item) {
144     switch (item.getItemId()) {
145         case android.R.id.home:
146             if (flgCount == 9) {
147                 flgCount = 0;
148                 moveActivity("com.example.motionauth", "com.example.motionauth.
149                     ViewDataList.ViewRegisteredRData", true);
150             }
151             else {
152                 flgCount++;
153             }
154             return true;
155         }
156     return false;
157 }
158
159 private void moveActivity (String pkgName, String actName, boolean flg) {
160     LogUtil.log(Log.INFO);
161
162     Intent intent = new Intent();
163     intent.setClassName(pkgName, actName);
```

```
164
165     intent.putExtra("item", item);
166
167     if (flg) intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.
        FLAG_ACTIVITY_NEW_TASK);
168
169     startActivityForResult(intent, 0);
170 }
171 }
```

C.8 src/com/example/motionauth/ViewDataList/ViewRegisteredRData.java

```
1 package com.example.motionauth.ViewDataList;
2
3 import android.app.ActionBar;
4 import android.app.Activity;
5 import android.content.Intent;
6 import android.os.Build;
7 import android.os.Bundle;
8 import android.os.Environment;
9 import android.util.Log;
10 import android.view.MenuItem;
11 import android.widget.AdapterView;
12 import android.widget.ListView;
13 import com.example.motionauth.R;
14 import com.example.motionauth.Utility.LogUtil;
15
16 import java.io.*;
17 import java.util.ArrayList;
18
19
20 /**
21  * モーション登録時の相関係数の結果を表示する
22  *
23  * @author Kensuke Kousaka
24  */
25 public class ViewRegisteredRData extends Activity {
26     String item = null;
27     int flgCount;
28
29 }
```



```
30  @Override
31  protected void onCreate (Bundle savedInstanceState) {
32      super.onCreate(savedInstanceState);
33
34      LogUtil.log(Log.INFO);
35
36      setContentView(R.layout.activity_view_registered_rdata);
37
38      if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.ICE_CREAM_SANDWICH) {
39          ActionBar actionBar = getActionBar();
40          if (actionBar != null) actionBar.setHomeButtonEnabled(true);
41      }
42      flgCount = 0;
43
44      viewRegisteredData();
45  }
46
47
48  private void viewRegisteredData () {
49      LogUtil.log(Log.INFO);
50
51      Intent intent = getIntent();
52      item = intent.getStringExtra("item");
53
54      ListView lv = (ListView) findViewById(R.id.listView1);
55
56      ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.
          simple_list_item_1);
57
58      ArrayList<String> dataList = readData();
59
60      for (String i : dataList) adapter.add(i);
61
62      lv.setAdapter(adapter);
63  }
64
65
66  private ArrayList<String> readData () {
67      LogUtil.log(Log.INFO);
68      ArrayList<String> dataList = new ArrayList<>();
69  }
```

```
70 String directoryPath = Environment.getExternalStorageDirectory().getPath()
    + File.separator + "MotionAuth" + File.separator + "RegistLRdata" + File
    .separator + item;
71
72 File directory = new File(directoryPath);
73 File[] fileList = directory.listFiles();
74
75 for (File aFileList : fileList) {
76     String filePath = directoryPath + File.separator + aFileList.getName();
77
78     File file = new File(filePath);
79
80     try {
81         FileInputStream fis = new FileInputStream(file);
82         InputStreamReader isr = new InputStreamReader(fis, "UTF-8");
83         BufferedReader br = new BufferedReader(isr);
84         String s;
85
86         while ((s = br.readLine()) != null) dataList.add(s);
87
88         br.close();
89         isr.close();
90         fis.close();
91     }
92     catch (FileNotFoundException e) {
93         e.printStackTrace();
94         return dataList;
95     }
96     catch (UnsupportedEncodingException e) {
97         e.printStackTrace();
98         return dataList;
99     }
100    catch (IOException e) {
101        e.printStackTrace();
102        return dataList;
103    }
104 }
105 return dataList;
106 }
107
108
```

```
109  @Override
110  public boolean onOptionsItemSelected (MenuItem item) {
111      switch (item.getItemId()) {
112          case android.R.id.home:
113              if (flgCount == 9) {
114                  flgCount = 0;
115                  moveActivity("com.example.motionauth", "com.example.motionauth.
                      ViewDataList.ViewAuthRData", true);
116              }
117              else {
118                  flgCount++;
119              }
120              return true;
121      }
122      return false;
123  }
124
125
126  private void moveActivity (String pkgName, String actName, boolean flg) {
127      LogUtil.log(Log.INFO);
128
129      Intent intent = new Intent();
130      intent.setClassName(pkgName, actName);
131
132      intent.putExtra("item", item);
133
134      if (flg) intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.
          FLAG_ACTIVITY_NEW_TASK);
135
136      startActivityForResult(intent, 0);
137  }
138 }
```

C.9 src/com/example/motionauth/ViewDataList/ViewAuthRData.java

```
1  package com.example.motionauth.ViewDataList;
2
3  import android.app.Activity;
4  import android.content.Intent;
5  import android.os.Bundle;
6  import android.os.Environment;
```

```
7 import android.util.Log;
8 import android.widget.AdapterView;
9 import android.widget.ListView;
10 import com.example.motionauth.R;
11 import com.example.motionauth.Utility.LogUtil;
12
13 import java.io.*;
14 import java.util.ArrayList;
15
16 /**
17  * 認証試験モードにおいて出た相関係数の結果を表示する
18  *
19  * @author Kensuke Kousaka
20  */
21 public class ViewAuthRData extends Activity {
22     String item = null;
23
24
25     @Override
26     protected void onCreate (Bundle savedInstanceState) {
27         super.onCreate(savedInstanceState);
28
29         LogUtil.log(Log.INFO);
30
31         setContentView(R.layout.activity_view_auth_rdata);
32
33         viewAuthRData();
34     }
35
36
37     private void viewAuthRData () {
38         LogUtil.log(Log.INFO);
39
40         Intent intent = getIntent();
41         item = intent.getStringExtra("item");
42
43         ListView lv = (ListView) findViewById(R.id.listView1);
44
45         ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.
            simple_list_item_1);
46
```

```
47     ArrayList<String> dataList = readData();
48
49     for (String i : dataList) adapter.add(i);
50
51     lv.setAdapter(adapter);
52 }
53
54
55 private ArrayList<String> readData () {
56     LogUtil.log(Log.INFO);
57     ArrayList<String> dataList = new ArrayList<>();
58
59     String filePath = Environment.getExternalStorageDirectory().getPath() +
        File.separator + "MotionAuth" + File.separator + "AuthRData" + File.
        separator + item;
60
61     File file = new File(filePath);
62
63     try {
64         FileInputStream fis = new FileInputStream(file);
65         InputStreamReader isr = new InputStreamReader(fis, "UTF-8");
66         BufferedReader br = new BufferedReader(isr);
67         String s;
68
69         while ((s = br.readLine()) != null) dataList.add(s);
70
71         br.close();
72         isr.close();
73         fis.close();
74     }
75     catch (FileNotFoundException e) {
76         e.printStackTrace();
77         return dataList;
78     }
79     catch (UnsupportedEncodingException e) {
80         e.printStackTrace();
81         return dataList;
82     }
83     catch (IOException e) {
84         e.printStackTrace();
85         return dataList;
```

```
86     }
87     return dataList;
88 }
89 }
```

C.10 src/com/example/motionauth/Processing/Formatter.java

```
1  package com.example.motionauth.Processing;
2
3  import android.util.Log;
4  import com.example.motionauth.Utility.LogUtil;
5
6  import java.util.Locale;
7
8
9  /**
10   * データの桁揃えを行う。対応データはfloat及びdouble。返り値はdouble型。
11   *
12   * @author Kensuke Kousaka
13   */
14  public class Formatter {
15
16      /**
17       * float型の2次元数値データを小数点以下2桁に揃え、doubleに変換する
18       *
19       * @param inputVal float型の2次元配列データ
20       * @return 小数点以下2桁に揃え、double型に変換した2次元数値データ
21       */
22      public double[][] floatToDoubleFormatter (float[][] inputVal) {
23          LogUtil.log(Log.INFO);
24
25          double[][] returnVal = new double[inputVal.length][inputVal[0].length];
26
27          for (int i = 0; i < inputVal.length; i++) {
28              for (int j = 0; j < inputVal[i].length; j++) {
29                  String format = String.format(Locale.getDefault(), "%.2f", inputVal[i][j]);
30                  returnVal[i][j] = Double.valueOf(format);
31              }
32          }
33      }
```

```
34     return returnVal;
35 }
36
37
38 /**
39  * float型の3次元数値データを小数点以下2桁に揃え、doubleに変換する
40  *
41  * @param inputVal float型の3次元配列データ
42  * @return 小数点以下2桁に揃え、double型に変換した3次元数値データ
43  */
44 public double[][][] floatToDoubleFormatter (float[][][] inputVal) {
45     LogUtil.log(Log.INFO);
46
47     double[][][] returnVal = new double[inputVal.length][inputVal[0].length][
48         inputVal[0][0].length];
49
50     for (int i = 0; i < inputVal.length; i++) {
51         for (int j = 0; j < inputVal[i].length; j++) {
52             for (int k = 0; k < inputVal[i][j].length; k++) {
53                 String format = String.format(Locale.getDefault(), "%.2f", inputVal[i]
54                     [j][k]);
55                 returnVal[i][j][k] = Double.valueOf(format);
56             }
57         }
58     }
59     return returnVal;
60 }
61
62 /**
63  * double型の二次元数値データを小数点以下二桁に揃える
64  *
65  * @param inputVal double型の二次元配列データ
66  * @return 小数点以下二桁に揃えたdouble型二次元数値データ
67  */
68 public double[][] doubleToDoubleFormatter (double[][] inputVal) {
69     LogUtil.log(Log.INFO);
70
71     double[][] returnVal = new double[inputVal.length][inputVal[0].length];
72 }
```

```

73     for (int i = 0; i < inputVal.length; i++) {
74         for (int j = 0; j < inputVal[i].length; j++) {
75             String format = String.format(Locale.getDefault(), "%.2f", inputVal[i][
                j]);
76             returnVal[i][j] = Double.valueOf(format);
77         }
78     }
79
80     return returnVal;
81 }
82
83
84 /**
85  * double型の3次元数値データを小数点以下2桁に揃える
86  *
87  * @param inputVal double型の3次元配列データ
88  * @return 小数点以下2桁に揃えたdouble型3次元数値データ
89  */
90 public double[][][] doubleToDoubleFormatter (double[][][] inputVal) {
91     LogUtil.log(Log.INFO);
92
93     double[][][] returnVal = new double[inputVal.length][inputVal[0].length][
        inputVal[0][0].length];
94
95     for (int i = 0; i < inputVal.length; i++) {
96         for (int j = 0; j < inputVal[i].length; j++) {
97             for (int k = 0; k < inputVal[i][j].length; k++) {
98                 String format = String.format(Locale.getDefault(), "%.2f", inputVal[i
                    ][j][k]);
99                 returnVal[i][j][k] = Double.valueOf(format);
100             }
101         }
102     }
103
104     return returnVal;
105 }
106 }

```

C.11 src/com/example/motionauth/Processing/Amplifier.java

```

1 package com.example.motionauth.Processing;

```



```
2
3 import android.util.Log;
4 import com.example.motionauth.Utility.LogUtil;
5
6 /**
7  * データの値を増幅させる
8  *
9  * @author Kensuke Kousaka
10 */
11 public class Amplifier {
12     private boolean isRangeCheck = false;
13
14
15     /**
16     * 全試行回数中、一回でもデータの幅が閾値よりも小さければtrueを返す
17     *
18     * @param data チェックするdouble型三次元配列データ
19     * @return 全試行回数中、一回でもデータの幅が閾値よりも小さければ
20         true、そうでなければfalse
21     */
22     public boolean CheckValueRange (double[][][] data, double checkRangeValue) {
23         LogUtil.log(Log.INFO);
24
25         LogUtil.log(Log.DEBUG, "checkRangeValue" + checkRangeValue);
26
27         double[][] max = new double[data.length][data[0].length];
28         double[][] min = new double[data.length][data[0].length];
29
30         for (int i = 0; i < data.length; i++) {
31             for (int j = 0; j < data[i].length; j++) {
32                 max[i][j] = 0;
33                 min[i][j] = 0;
34             }
35         }
36
37         double range;
38         for (int i = 0; i < data.length; i++) {
39             for (int j = 0; j < data[i].length; j++) {
40                 for (int k = 0; k < data[i][j].length; k++) {
41                     if (data[i][j][k] > max[i][j]) {
42                         max[i][j] = data[i][j][k];
43                     }
44                     if (data[i][j][k] < min[i][j]) {
45                         min[i][j] = data[i][j][k];
46                     }
47                 }
48             }
49         }
50         range = (max[0][0] - min[0][0]) / 2;
51         return range > checkRangeValue;
52     }
53 }
```

```
42     }
43     else if (data[i][j][k] < min[i][j]) {
44         min[i][j] = data[i][j][k];
45     }
46 }
47 }
48 }
49
50 for (int i = 0; i < max.length; i++) {
51     for (int j = 0; j < max[i].length; j++) {
52         range = max[i][j] - min[i][j];
53         LogUtil.log(Log.DEBUG, "range = " + range);
54         if (range < checkRangeValue) isRangeCheck = true;
55     }
56 }
57
58 return isRangeCheck;
59 }
60
61
62 /**
63  * 与えられたデータを増幅させる
64  *
65  * @param data 増幅させる double 型三次元配列データ
66  * @param ampValue どれだけデータを増幅させるか
67  * @return 増幅後の double 型三次元配列データ
68  */
69 public double[][][] Amplify (double[][][] data, double ampValue) {
70     LogUtil.log(Log.INFO);
71
72     if (ampValue != 0.0) {
73         for (int i = 0; i < data.length; i++) {
74             for (int j = 0; j < data[i].length; j++) {
75                 for (int k = 0; k < data[i][j].length; k++) {
76                     data[i][j][k] *= ampValue;
77                 }
78             }
79         }
80     }
81     return data;
82 }
```

```
83
84
85  /**
86   * 与えられたデータを増幅させる
87   *
88   * @param data 増幅させる double 型二次元配列データ
89   * @param ampValue どれだけデータを増幅させるか
90   * @return 増幅後の double 型二次元配列データ
91   */
92  public double[][] Amplify (double[][] data, double ampValue) {
93      LogUtil.log(Log.INFO);
94
95      if (ampValue != 0.0) {
96          for (int i = 0; i < data.length; i++) {
97              for (int j = 0; j < data[i].length; j++) {
98                  data[i][j] *= ampValue;
99              }
100          }
101      }
102
103      return data;
104  }
105 }
```

C.12 src/com/example/motionauth/Processing/Calc.java

```
1  package com.example.motionauth.Processing;
2
3  import android.util.Log;
4  import com.example.motionauth.Utility.LogUtil;
5
6  /**
7   * 加速度や角速度から、速度や角度を求める
8   *
9   * @author Kensuke Kousaka
10  */
11  public class Calc {
12
13      /**
14       * 加速度データを距離データに変換する
15       *
```

```
16  * @param inputVal 変換対象の , 三次元加速度データ
17  * @param t        時間
18  * @return 変換後の三次元距離データ
19  */
20  public double[][][] accelToDistance (double[][][] inputVal, double t) {
21      LogUtil.log(Log.INFO);
22
23      double[][][] returnVal = new double[inputVal.length][inputVal[0].length][
        inputVal[0][0].length];
24
25      for (int i = 0; i < inputVal.length; i++) {
26          for (int j = 0; j < inputVal[i].length; j++) {
27              for (int k = 0; k < inputVal[i][j].length; k++) {
28                  returnVal[i][j][k] = (inputVal[i][j][k] * t * t) / 2 * 1000;
29              }
30          }
31      }
32
33      return returnVal;
34  }
35
36
37  /**
38   * 角速度データを角度データに変換する
39   *
40   * @param inputVal 変換対象の , 三次元角速度データ
41   * @param t        時間
42   * @return 変換後の三次元角度データ
43   */
44  public double[][][] gyroToAngle (double[][][] inputVal, double t) {
45      LogUtil.log(Log.INFO);
46
47      double[][][] returnVal = new double[inputVal.length][inputVal[0].length][
        inputVal[0][0].length];
48
49      for (int i = 0; i < inputVal.length; i++) {
50          for (int j = 0; j < inputVal[i].length; j++) {
51              for (int k = 0; k < inputVal[i][j].length; k++) {
52                  returnVal[i][j][k] = (inputVal[i][j][k] * t) * 1000;
53              }
54          }
55      }
```

```
55     }
56
57     return returnVal;
58 }
59
60
61 /**
62  * 加速度データを距離データに変換する
63  *
64  * @param inputVal 変換対象の, 三次元加速度データ
65  * @param t        時間
66  * @return 変換後の三次元距離データ
67  */
68 public double[][] accelToDistance (double[][] inputVal, double t) {
69     LogUtil.log(Log.INFO);
70
71     double[][] returnVal = new double[inputVal.length][inputVal[0].length];
72
73     for (int i = 0; i < inputVal.length; i++) {
74         for (int j = 0; j < inputVal[i].length; j++) {
75             returnVal[i][j] = (inputVal[i][j] * t * t) / 2 * 1000;
76         }
77     }
78
79     return returnVal;
80 }
81
82
83 /**
84  * 角速度データを角度データに変換する
85  *
86  * @param inputVal 変換対象の, 三次元角速度データ
87  * @param t        時間
88  * @return 変換後の三次元角度データ
89  */
90 public double[][] gyroToAngle (double[][] inputVal, double t) {
91     LogUtil.log(Log.INFO);
92
93     double[][] returnVal = new double[inputVal.length][inputVal[0].length];
94
95     for (int i = 0; i < inputVal.length; i++) {
```

```

96     for (int j = 0; j < inputVal[i].length; j++) {
97         returnVal[i][j] = (inputVal[i][j] * t) * 1000;
98     }
99 }
100
101 return returnVal;
102 }
103 }

```

C.13 src/com/example/motionauth/Processing/Correlation.java

```

1 package com.example.motionauth.Processing;
2
3 import android.util.Log;
4 import com.example.motionauth.Authentication.AuthNameInput;
5 import com.example.motionauth.Registration.RegistNameInput;
6 import com.example.motionauth.Utility.Enum;
7 import com.example.motionauth.Utility.LogUtil;
8 import com.example.motionauth.Utility.ManageData;
9
10
11 /**
12  * 相関を求める
13  *
14  * @author Kensuke Kousaka
15  */
16 public class Correlation {
17     private ManageData mManageData = new ManageData();
18     private Enum mEnum = new Enum();
19
20     /**
21      * 相関を求め、同一のモーションであるかどうかを確認する
22      *
23      * @param distance double型の3次元配列距離データ
24      * @param angle double型の3次元配列角度データ
25      * @param ave_distance double型の二次元配列距離データ
26      * @param ave_angle double型の二次元配列角度データ
27      * @return EnumクラスのMEASURE列挙体の値が返る
28      */
29     public Enum.MEASURE measureCorrelation (double[][][] distance, double[][][]
        angle, double[][] ave_distance, double[][] ave_angle) {

```

```
30 LogUtil.log(Log.INFO);
31
32 // 相関係数の計算
33
34 // Calculate of Average A
35 float[][] sample_accel = new float[3][3];
36 float[][] sample_gyro = new float[3][3];
37
38 // iは回数
39 for (int i = 0; i < 3; i++) {
40     // jはXYZ
41     for (int j = 0; j < 3; j++) {
42         for (int k = 0; k < 100; k++) {
43             sample_accel[i][j] += distance[i][j][k];
44             sample_gyro[i][j] += angle[i][j][k];
45         }
46     }
47 }
48
49 for (int i = 0; i < 3; i++) {
50     for (int j = 0; j < 3; j++) {
51         sample_accel[i][j] /= 99;
52         sample_gyro[i][j] /= 99;
53     }
54 }
55
56 // Calculate of Average B
57 float ave_accel[] = new float[3];
58 float ave_gyro[] = new float[3];
59
60 for (int i = 0; i < 3; i++) {
61     for (int j = 0; j < 100; j++) {
62         ave_accel[i] += ave_distance[i][j];
63         ave_gyro[i] += ave_angle[i][j];
64     }
65 }
66
67 for (int i = 0; i < 3; i++) {
68     ave_accel[i] /= 99;
69     ave_gyro[i] /= 99;
70 }
```

```
71
72 // Calculate of Sxx
73 float Sxx_accel[][] = new float[3][3];
74 float Sxx_gyro[][] = new float[3][3];
75
76 for (int i = 0; i < 3; i++) {
77     for (int j = 0; j < 3; j++) {
78         for (int k = 0; k < 100; k++) {
79             Sxx_accel[i][j] += Math.pow((distance[i][j][k] - sample_accel[i][j]),
80                                     2);
81             Sxx_gyro[i][j] += Math.pow((angle[i][j][k] - sample_gyro[i][j]), 2);
82         }
83     }
84 }
85
86 // Calculate of Syy
87 float Syy_accel[] = new float[3];
88 float Syy_gyro[] = new float[3];
89
90 for (int i = 0; i < 3; i++) {
91     for (int j = 0; j < 100; j++) {
92         Syy_accel[i] += Math.pow((ave_distance[i][j] - ave_accel[i]), 2);
93         Syy_gyro[i] += Math.pow((ave_angle[i][j] - ave_gyro[i]), 2);
94     }
95 }
96
97 // Calculate of Sxy
98 float[][] Sxy_accel = new float[3][3];
99 float[][] Sxy_gyro = new float[3][3];
100
101 for (int i = 0; i < 3; i++) {
102     for (int j = 0; j < 3; j++) {
103         for (int k = 0; k < 100; k++) {
104             Sxy_accel[i][j] += (distance[i][j][k] - sample_accel[i][j]) * (
105                 ave_distance[j][k] - ave_accel[j]);
106             Sxy_gyro[i][j] += (angle[i][j][k] - sample_gyro[i][j]) * (ave_angle[j]
107                 [k] - ave_gyro[j]);
108         }
109     }
110 }
```



```
109 // Calculate of R
110 double[][] R_accel = new double[3][3];
111 double[][] R_gyro = new double[3][3];
112
113 for (int i = 0; i < 3; i++) {
114     for (int j = 0; j < 3; j++) {
115         R_accel[i][j] = Sxy_accel[i][j] / Math.sqrt(Sxx_accel[i][j] * Syy_accel
116             [j]);
117         R_gyro[i][j] = Sxy_gyro[i][j] / Math.sqrt(Sxx_gyro[i][j] * Syy_gyro[j
118             l]);
119     }
120 }
121
122 mManageData.writeRData("RegistLRdata", "R_accel", RegistNameInput.name,
123     R_accel);
124 mManageData.writeRData("RegistLRdata", "R_gyro", RegistNameInput.name,
125     R_gyro);
126
127 for (double[] i : R_accel) {
128     for (double j : i) {
129         LogUtil.log(Log.DEBUG, "R_accel: " + j);
130     }
131 }
132
133 for (double[] i : R_gyro) {
134     for (double j : i) {
135         LogUtil.log(Log.DEBUG, "R_gyro: " + j);
136     }
137 }
138
139 double R_point = 0.0;
140 for (double[] i : R_accel) {
141     for (double j : i) {
142         R_point += j;
143     }
144 }
145 for (double[] i : R_gyro) {
146     for (double j : i) {
147         R_point += j;
148     }
149 }
```

```

146
147     R_point = R_point / 18;
148
149     mManageData.writeRpoint("Rpoint", RegistNameInput.name, R_point);
150
151     // X
152     if ((R_accel[0][0] > mEnum.LOOSE && R_accel[1][0] > mEnum.LOOSE) || (
        R_accel[1][0] > mEnum.LOOSE && R_accel[2][0] > mEnum.LOOSE) || (R_accel
        [0][0] > mEnum.LOOSE && R_accel[2][0] > mEnum.LOOSE)) {
153     // Y
154     if ((R_accel[0][1] > mEnum.LOOSE && R_accel[1][1] > mEnum.LOOSE) || (
        R_accel[1][1] > mEnum.LOOSE && R_accel[2][1] > mEnum.LOOSE) || (
        R_accel[0][1] > mEnum.LOOSE && R_accel[2][1] > mEnum.LOOSE)) {
155     // Z
156     if ((R_accel[0][2] > mEnum.LOOSE && R_accel[1][2] > mEnum.LOOSE) || (
        R_accel[1][2] > mEnum.LOOSE && R_accel[2][2] > mEnum.LOOSE) || (
        R_accel[0][2] > mEnum.LOOSE && R_accel[2][2] > mEnum.LOOSE)) {
157     // X
158     if ((R_gyro[0][0] > mEnum.LOOSE && R_gyro[1][0] > mEnum.LOOSE) || (
        R_gyro[1][0] > mEnum.LOOSE && R_gyro[2][0] > mEnum.LOOSE) || (
        R_gyro[0][0] > mEnum.LOOSE || R_gyro[2][0] > mEnum.LOOSE)) {
159     // Y
160     if ((R_gyro[0][1] > mEnum.LOOSE && R_gyro[1][1] > mEnum.LOOSE) || (
        R_gyro[1][1] > mEnum.LOOSE && R_gyro[2][1] > mEnum.LOOSE) || (
        R_gyro[0][1] > mEnum.LOOSE || R_gyro[2][1] > mEnum.LOOSE)) {
161     // Z
162     if ((R_gyro[0][2] > mEnum.LOOSE && R_gyro[1][2] > mEnum.LOOSE) ||
        (R_gyro[1][2] > mEnum.LOOSE && R_gyro[2][2] > mEnum.LOOSE) ||
        (R_gyro[0][2] > mEnum.LOOSE && R_gyro[2][2] > mEnum.LOOSE)) {
163
164     // X
165     if ((R_accel[0][0] > mEnum.NORMAL && R_accel[1][0] > mEnum.
        NORMAL) || (R_accel[1][0] > mEnum.NORMAL && R_accel[2][0] >
        mEnum.NORMAL) || (R_accel[0][0] > mEnum.NORMAL && R_accel
        [2][0] > mEnum.NORMAL)) {
166     // Y
167     if ((R_accel[0][1] > mEnum.NORMAL && R_accel[1][1] > mEnum.
        NORMAL) || (R_accel[1][1] > mEnum.NORMAL && R_accel[2][1]
        > mEnum.NORMAL) || (R_accel[0][1] > mEnum.NORMAL &&
        R_accel[2][1] > mEnum.NORMAL)) {
168     // Z

```

```

169         if ((R_accel[0][2] > mEnum.NORMAL && R_accel[1][2] > mEnum.
NORMAL) || (R_accel[1][2] > mEnum.NORMAL && R_accel
[2][2] > mEnum.NORMAL) || (R_accel[0][2] > mEnum.NORMAL
&& R_accel[2][2] > mEnum.NORMAL)) {
170             // X
171         if ((R_gyro[0][0] > mEnum.NORMAL && R_gyro[1][0] > mEnum.
NORMAL) || (R_gyro[1][0] > mEnum.NORMAL && R_gyro
[2][0] > mEnum.NORMAL) || (R_gyro[0][0] > mEnum.NORMAL
&& R_gyro[2][0] > mEnum.NORMAL)) {
172             // Y
173         if ((R_gyro[0][1] > mEnum.NORMAL && R_gyro[1][1] >
mEnum.NORMAL) || (R_gyro[1][1] > mEnum.NORMAL &&
R_gyro[2][1] > mEnum.NORMAL) || (R_gyro[0][1] >
mEnum.NORMAL && R_gyro[2][1] > mEnum.NORMAL)) {
174             // Z
175         if ((R_gyro[0][2] > mEnum.NORMAL && R_gyro[1][2] >
mEnum.NORMAL) || (R_gyro[1][2] > mEnum.NORMAL &&
R_gyro[2][2] > mEnum.NORMAL) || (R_gyro[0][2] >
mEnum.NORMAL && R_gyro[2][2] > mEnum.NORMAL)) {
176
177             // X
178         if ((R_accel[0][0] > mEnum.STRICT && R_accel[1][0]
> mEnum.STRICT) || (R_accel[1][0] > mEnum.STRICT
&& R_accel[2][0] > mEnum.STRICT) || (R_accel
[0][0] > mEnum.STRICT && R_accel[2][0] > mEnum.
STRICT)) {
179             // Y
180         if ((R_accel[0][1] > mEnum.STRICT && R_accel
[1][1] > mEnum.STRICT) || (R_accel[1][1] >
mEnum.STRICT && R_accel[2][1] > mEnum.STRICT)
|| (R_accel[0][1] > mEnum.STRICT && R_accel
[2][1] > mEnum.STRICT)) {
181             // Z
182         if ((R_accel[0][2] > mEnum.STRICT && R_accel
[1][2] > mEnum.STRICT) || (R_accel[1][2] >
mEnum.STRICT && R_accel[2][2] > mEnum.STRICT
) || (R_accel[0][2] > mEnum.STRICT &&
R_accel[2][2] > mEnum.STRICT)) {
183             // X
184         if ((R_gyro[0][0] > mEnum.STRICT && R_gyro
[1][0] > mEnum.STRICT) || (R_gyro[1][0] >

```

```

185         mEnum.STRICT && R_gyro[2][0] > mEnum.
186         STRICT) || (R_gyro[0][0] > mEnum.STRICT &&
            R_gyro[2][0] > mEnum.STRICT)) {
187         // Y
188         if ((R_gyro[0][1] > mEnum.STRICT && R_gyro
            [1][1] > mEnum.STRICT) || (R_gyro[1][1]
            > mEnum.STRICT && R_gyro[2][1] > mEnum.
            STRICT) || (R_gyro[0][1] > mEnum.STRICT
            && R_gyro[2][1] > mEnum.STRICT)) {
189         // Z
190         if ((R_gyro[0][2] > mEnum.STRICT &&
            R_gyro[1][2] > mEnum.STRICT) || (
            R_gyro[1][2] > mEnum.STRICT && R_gyro
            [2][2] > mEnum.STRICT) || (R_gyro
            [0][2] > mEnum.STRICT && R_gyro[2][2]
            > mEnum.STRICT)) {
191             return Enum.MEASURE.PERFECT;
192         }
193         // NORMALより大きくSTRICT以下
194         else {
195             return Enum.MEASURE.CORRECT;
196         }
197     }
198     else {
199         return Enum.MEASURE.CORRECT;
200     }
201     else {
202         return Enum.MEASURE.CORRECT;
203     }
204 }
205 }
206 else {
207     return Enum.MEASURE.CORRECT;
208 }
209 }
210 else {
211     return Enum.MEASURE.CORRECT;
212 }
213 }

```

```
214         else {
215             return Enum.MEASURE.CORRECT;
216         }
217
218     }
219     // LOOSEより大きくNORMAL以下
220     else {
221         return Enum.MEASURE.INCORRECT;
222     }
223 }
224 else {
225     return Enum.MEASURE.INCORRECT;
226 }
227
228 }
229 else {
230     return Enum.MEASURE.INCORRECT;
231 }
232
233 }
234 else {
235     return Enum.MEASURE.INCORRECT;
236 }
237 }
238 else {
239     return Enum.MEASURE.INCORRECT;
240 }
241 }
242 else {
243     return Enum.MEASURE.INCORRECT;
244 }
245 }
246 // LOOSE以下
247 else {
248     return Enum.MEASURE.BAD;
249 }
250 }
251 else {
252     return Enum.MEASURE.BAD;
253 }
254 }
```

```
255         else {
256             return Enum.MEASURE.BAD;
257         }
258     }
259     else {
260         return Enum.MEASURE.BAD;
261     }
262 }
263 else {
264     return Enum.MEASURE.BAD;
265 }
266 }
267 else {
268     return Enum.MEASURE.BAD;
269 }
270 }
271
272
273 /**
274  * 相関を求め、同一のモーションであるかどうかを確認する
275  *
276  * @param distance    double型の二次元配列距離データ
277  * @param angle       double型の二次元配列角度データ
278  * @param ave_distance double型の二次元配列距離データ
279  * @param ave_angle   double型の虹連配列角度データ
280  * @return EnumクラスのMEASURE列挙体の値が返る
281  */
282 public Enum.MEASURE measureCorrelation (double[][] distance, double[][] angle
283     , double[][] ave_distance, double[][] ave_angle) {
284     LogUtil.log(Log.INFO);
285
286     LogUtil.log(Log.DEBUG, "distancesample: " + String.valueOf(distance
287         [0][0]));
288     LogUtil.log(Log.DEBUG, "anglesample: " + String.valueOf(angle[0][0]));
289     LogUtil.log(Log.DEBUG, "avedistancesample: " + String.valueOf(ave_distance
290         [0][0]));
291     LogUtil.log(Log.DEBUG, "aveanglesample: " + String.valueOf(ave_angle
292         [0][0]));
293
294     //region Calculate of Average A
295     float[] sample_accel = new float[3];
```

```
292     float[] sample_gyro = new float[3];
293
294     for (int i = 0; i < 3; i++) {
295         for (int j = 0; j < 100; j++) {
296             sample_accel[i] += distance[i][j];
297             sample_gyro[i] += angle[i][j];
298         }
299     }
300
301     for (int i = 0; i < 3; i++) {
302         sample_accel[i] /= 99;
303         sample_gyro[i] /= 99;
304     }
305     //endregion
306
307     //region Calculate of Average B
308     float ave_accel[] = new float[3];
309     float ave_gyro[] = new float[3];
310
311     for (int i = 0; i < 3; i++) {
312         for (int j = 0; j < 100; j++) {
313             ave_accel[i] += ave_distance[i][j];
314             ave_gyro[i] += ave_angle[i][j];
315         }
316     }
317
318     for (int i = 0; i < 3; i++) {
319         ave_accel[i] /= 99;
320         ave_gyro[i] /= 99;
321     }
322     //endregion
323
324     //region Calculate of Sxx
325     float Sxx_accel[] = new float[3];
326     float Sxx_gyro[] = new float[3];
327
328     for (int i = 0; i < 3; i++) {
329         for (int j = 0; j < 100; j++) {
330             Sxx_accel[i] += Math.pow((distance[i][j] - sample_accel[i]), 2);
331             Sxx_gyro[i] += Math.pow((angle[i][j] - sample_gyro[i]), 2);
332         }
```

```

333     }
334     //endregion
335
336     //region Calculate of Syy
337     float Syy_accel[] = new float[3];
338     float Syy_gyro[] = new float[3];
339
340     for (int i = 0; i < 3; i++) {
341         for (int j = 0; j < 100; j++) {
342             Syy_accel[i] += Math.pow((ave_distance[i][j] - ave_accel[i]), 2);
343             Syy_gyro[i] += Math.pow((ave_angle[i][j] - ave_gyro[i]), 2);
344         }
345     }
346     //endregion
347
348     //region Calculate of Sxy
349     float Sxy_accel[] = new float[3];
350     float Sxy_gyro[] = new float[3];
351
352     for (int i = 0; i < 3; i++) {
353         for (int j = 0; j < 100; j++) {
354             Sxy_accel[i] += (distance[i][j] - sample_accel[i]) * (ave_distance[i][j]
355                 - ave_accel[i]);
356             Sxy_gyro[i] += (angle[i][j] - sample_gyro[i]) * (ave_angle[i][j] -
357                 ave_gyro[i]);
358         }
359     }
360     //endregion
361
362     //region Calculate of R
363     double R_accel[] = new double[3];
364     double R_gyro[] = new double[3];
365
366     for (int i = 0; i < 3; i++) {
367         R_accel[i] = Sxy_accel[i] / Math.sqrt(Sxx_accel[i] * Syy_accel[i]);
368         LogUtil.log(Log.DEBUG, "R_accel" + i + ": " + R_accel[i]);
369         R_gyro[i] = Sxy_gyro[i] / Math.sqrt(Sxx_gyro[i] * Syy_gyro[i]);
370         LogUtil.log(Log.DEBUG, "R_gyro" + i + ": " + R_gyro[i]);
371     }
372     //endregion

```



```
372 mManageData.writeRData("AuthRData", AuthNameInput.name, R_accel, R_gyro);
373
374 //region  相 関 の 判 定
375 //相関係数が一定以上あるなら認証成功
376 if (R_accel[0] > 0.5) {
377     if (R_accel[1] > 0.5) {
378         if (R_accel[2] > 0.5) {
379             if (R_gyro[0] > 0.5) {
380                 if (R_gyro[1] > 0.5) {
381                     if (R_gyro[2] > 0.5) {
382                         return Enum.MEASURE.CORRECT;
383                     }
384                     else {
385                         return Enum.MEASURE.INCORRECT;
386                     }
387                 }
388                 else {
389                     return Enum.MEASURE.INCORRECT;
390                 }
391             }
392             else {
393                 return Enum.MEASURE.INCORRECT;
394             }
395         }
396         else {
397             return Enum.MEASURE.INCORRECT;
398         }
399     }
400     else {
401         return Enum.MEASURE.INCORRECT;
402     }
403 }
404 else {
405     return Enum.MEASURE.INCORRECT;
406 }
407 //endregion
408
409 }
410 }
```

```
1 package com.example.motionauth.Processing;
2
3 import android.util.Log;
4 import com.example.motionauth.Utility.Enum;
5 import com.example.motionauth.Utility.LogUtil;
6
7 import java.util.ArrayList;
8 import java.util.Collections;
9 import java.util.TreeMap;
10
11
12 /**
13  * データの時間的なズレを修正する
14  *
15  * @author Kensuke Kousaka
16  */
17 public class CorrectDeviation {
18
19     /**
20      * 取得回数ごとのデータのズレを時間的なズレを修正する
21      *
22      * @param distance 修正する距離データ
23      * @param angle 修正する角度データ
24      * @param mode どこを基準にとるか
25      * @param target どちらのデータに基準を置くか
26      * @return newData ズレ修正後のdouble型の4次元配列データ
27      */
28     public double[][][][] correctDeviation (double[][][] distance, double[][][]
29         angle, Enum.MODE mode, Enum.TARGET target) {
30         LogUtil.log(Log.INFO);
31
32         // ずらしたデータを格納する配列
33         double[][][][] newData = new double[2][3][3][100];
34
35         // 試行回数ごとの代表値の出ている時間を抽出
36         // 変数は、桁揃え、計算後のdistance, angleを利用する
37
38         // 回数・XYZを配列で
39         double tmpValue[][] = new double[3][3];
40
41         // 代表値の出ている時間、回数、XYZ
```

```
41     int count[][] = new int[3][3];
42
43     // 変数に3回分XYZそれぞれの1個目の値を放り込む
44     switch (target) {
45         case DISTANCE:
46             for (int i = 0; i < 3; i++) {
47                 for (int j = 0; j < 3; j++) {
48                     tmpValue[i][j] = distance[i][j][0];
49                 }
50             }
51             break;
52         case ANGLE:
53             for (int i = 0; i < 3; i++) {
54                 for (int j = 0; j < 3; j++) {
55                     tmpValue[i][j] = angle[i][j][0];
56                 }
57             }
58             break;
59     }
60
61     // 代表値が出ている場所を取得する
62     switch (target) {
63         case DISTANCE:
64             switch (mode) {
65                 case MAX:
66                     for (int i = 0; i < 3; i++) {
67                         for (int j = 0; j < 3; j++) {
68                             for (int k = 0; k < 100; k++) {
69                                 if (tmpValue[i][j] < distance[i][j][k]) {
70                                     tmpValue[i][j] = distance[i][j][k];
71                                     count[i][j] = k;
72                                 }
73                             }
74                         }
75                     }
76                     break;
77                 case MIN:
78                     for (int i = 0; i < 3; i++) {
79                         for (int j = 0; j < 3; j++) {
80                             for (int k = 0; k < 100; k++) {
81                                 if (tmpValue[i][j] > distance[i][j][k]) {
```

```
82         tmpValue[i][j] = distance[i][j][k];
83         count[i][j] = k;
84     }
85 }
86 }
87 }
88 break;
89 case MEDIAN:
90     // キーが自動ソートされる TreeMap を用いる .
91     // データと順番を紐付けしたものを作成し , 中央値の初期の
92     // 順番の値を取り出す .
93     for (int i = 0; i < 3; i++) {
94         for (int j = 0; j < 3; j++) {
95             TreeMap<Double, Integer> treeMap = new TreeMap<>();
96
97             for (int k = 0; k < 100; k++) {
98                 treeMap.put(distance[i][j][k], k);
99             }
100
101             int loopCount = 0;
102             for (Integer initCount : treeMap.values()) {
103                 if (loopCount == 49) {
104                     count[i][j] = initCount;
105                 }
106                 loopCount++;
107             }
108         }
109     }
110     break;
111 }
112 break;
113 case ANGLE:
114     switch (mode) {
115     case MAX:
116         for (int i = 0; i < 3; i++) {
117             for (int j = 0; j < 3; j++) {
118                 for (int k = 0; k < 100; k++) {
119                     if (tmpValue[i][j] < angle[i][j][k]) {
120                         tmpValue[i][j] = angle[i][j][k];
121                         count[i][j] = k;
```

```
122         }
123     }
124 }
125 }
126 break;
127 case MIN:
128     for (int i = 0; i < 3; i++) {
129         for (int j = 0; j < 3; j++) {
130             for (int k = 0; k < 100; k++) {
131                 if (tmpValue[i][j] > angle[i][j][k]) {
132                     tmpValue[i][j] = angle[i][j][k];
133                     count[i][j] = k;
134                 }
135             }
136         }
137     }
138     break;
139 case MEDIAN:
140     // キーが自動ソートされる TreeMap を用いる .
141     // データと順番を紐付けしたものを作成し , 中央値の初期の
142     // 順番の値を取り出す .
143     for (int i = 0; i < 3; i++) {
144         for (int j = 0; j < 3; j++) {
145             TreeMap<Double, Integer> treeMap = new TreeMap<>();
146
147             for (int k = 0; k < 100; k++) {
148                 treeMap.put(angle[i][j][k], k);
149             }
150
151             int loopCount = 0;
152             for (Integer initCount : treeMap.values()) {
153                 if (loopCount == 49) {
154                     count[i][j] = initCount;
155                 }
156                 loopCount++;
157             }
158         }
159     }
160     break;
161 }
```

```
162         break;
163     }
164
165     // 1回目のデータの代表値が出た場所と、2回目・3回目のデータの代表値が出た場
    所の差を取る
166     // 取ったら、その差だけデータをずらす（ずらしてはみ出たデータは空いたところ
    に入れる）
167
168     // ずらす移動量を計算
169     int lagData[][] = new int[2][3];
170
171     // どれだけズレているかを計算する
172     for (int i = 0; i < 3; i++) {
173         lagData[0][i] = count[0][i] - count[1][i];
174         LogUtil.log(Log.DEBUG, "lagData[0]" + "[" + i + "]" + ": " + lagData[0][i]
            );
175
176         lagData[1][i] = count[0][i] - count[2][i];
177         LogUtil.log(Log.DEBUG, "lagData[1]" + "[" + i + "]" + ": " + lagData[1][i]
            );
178     }
179
180     // 1回目のデータに関しては基準となるデータなのでそのまま入れる
181     for (int i = 0; i < 3; i++) {
182         for (int j = 0; j < 100; j++) {
183             newData[0][0][i][j] = distance[0][i][j];
184             newData[1][0][i][j] = angle[0][i][j];
185         }
186     }
187
188     // 実際にデータをずらしていく（ずらすのは、1回目を除くデータ）
189     for (int i = 1; i < 3; i++) {
190         for (int j = 0; j < 3; j++) {
191             ArrayList<Double> distanceTemp = new ArrayList<>();
192             ArrayList<Double> angleTemp = new ArrayList<>();
193
194             for (int k = 0; k < 100; k++) {
195                 distanceTemp.add(distance[i][j][k]);
196                 angleTemp.add(angle[i][j][k]);
197             }
198             Collections.rotate(distanceTemp, lagData[i - 1][j]);
```

```
199     Collections.rotate(angleTemp, lagData[i - 1][j]);
200     for (int k = 0; k < 100; k++) {
201         newData[0][i][j][k] = distanceTemp.get(k);
202         newData[1][i][j][k] = angleTemp.get(k);
203     }
204 }
205 }
206
207 return newData;
208 }
209 }
```

C.15 src/com/example/motionauth/Processing/CipherCrypt.java

```
1 package com.example.motionauth.Processing;
2
3 import android.content.Context;
4 import android.content.SharedPreferences;
5 import android.util.Base64;
6 import android.util.Log;
7 import com.example.motionauth.Utility.LogUtil;
8
9 import javax.crypto.*;
10 import javax.crypto.spec.IvParameterSpec;
11 import javax.crypto.spec.SecretKeySpec;
12 import java.security.*;
13
14
15 /**
16  * 暗号化に関する処理
17  *
18  * @author Kensuke Kousaka
19  */
20 public class CipherCrypt {
21     private static final int ENCRYPT_KEY_LENGTH = 128;
22     private static final String PREF_KEY = "Cipher";
23     private static final String CIPHER_KEY = "CipherCrypt";
24     private static final String CIPHER_IV = "CipherIv";
25
26     private final Key key;
27     private final IvParameterSpec iv;
```

```
28
29
30 /**
31  * 暗号化・復号に必要な Secret Key , IV (Initialization Vector) の準備を行う
32  *
33  * @param context アプリケーション固有の
34  *   SharedPreferences を取得する際に用いる Context
35  */
36 public CipherCrypt (Context context) {
37     LogUtil.log(Log.INFO);
38
39     Context mContext = context.getApplicationContext();
40
41     // SharedPreferences を取得する
42     SharedPreferences preferences = mContext.getSharedPreferences(PREF_KEY,
43         Context.MODE_PRIVATE);
44     SharedPreferences.Editor editor = preferences.edit();
45
46     // Preference から Secret Key を取得 ( 値が保存されていなければ , 空文字を返す )
47     String keyStr = preferences.getString(CIPHER_KEY, "");
48
49     if ("".equals(keyStr)) {
50         LogUtil.log(Log.DEBUG, "Couldn't get cipher key from preferences");
51         // Preference から取得できなかった場合
52         // Secret Key を生成し , 保存する
53
54         // Secret Key を生成
55         key = generateKey();
56
57         // 生成した Secret Key を保存
58         String base64Key = Base64.encodeToString(key.getEncoded(), Base64.
59             URL_SAFE | Base64.NO_WRAP);
60
61         editor.putString(CIPHER_KEY, base64Key).apply();
62     }
63     else {
64         LogUtil.log(Log.DEBUG, "Get cipher key from preferences");
65         // Preference から取得できた場合
66         // Secret Key を復元
67         byte[] byteKey = Base64.decode(keyStr, Base64.URL_SAFE | Base64.NO_WRAP);
68         key = new SecretKeySpec(byteKey, "AES");
69     }
70 }
```



```
66     }
67
68
69     // PreferenceからIVを取得（値が保存されていなければ，空文字を返す）
70     String ivStr = preferences.getString(CIPHER_IV, "");
71
72     if ("".equals(ivStr)) {
73         LogUtil.log(Log.DEBUG, "Couldn't get iv from preferences");
74         // Preferenceから取得できなかった場合
75         // IVを生成し，保存する
76
77         // IVを生成
78         byte[] byteIv = generateIv();
79         iv = new IvParameterSpec(byteIv);
80
81         // 生成したIVを保存
82         String base64Iv = Base64.encodeToString(byteIv, Base64.URL_SAFE | Base64.NO_WRAP);
83
84         editor.putString(CIPHER_IV, base64Iv).apply();
85     }
86     else {
87         LogUtil.log(Log.DEBUG, "Get iv from preferences");
88         // Preferenceから取得できた場合
89         // IVを復元
90         byte[] byteIv = Base64.decode(ivStr, Base64.URL_SAFE | Base64.NO_WRAP);
91         iv = new IvParameterSpec(byteIv);
92     }
93 }
94
95
96 /**
97  * 暗号化・復号に使用する Secret Keyを生成する
98  *
99  * @return Secret Key
100 */
101 private Key generateKey () {
102     LogUtil.log(Log.INFO);
103     try {
104         KeyGenerator generator = KeyGenerator.getInstance("AES");
105         SecureRandom random = SecureRandom.getInstance("SHA1PRNG");
```

```
106
107     generator.init(ENCRYPT_KEY_LENGTH, random);
108
109     return generator.generateKey();
110 }
111 catch (NoSuchAlgorithmException e) {
112     throw new RuntimeException(e);
113 }
114 }
115
116
117 /**
118  * 暗号化・復号に使用するIVを生成する
119  *
120  * @return byte配列型のIV
121  */
122 private byte[] generateIv () {
123     LogUtil.log(Log.INFO);
124     try {
125         Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
126         cipher.init(Cipher.ENCRYPT_MODE, key);
127
128         return cipher.getIV();
129     }
130     catch (NoSuchAlgorithmException e) {
131         throw new RuntimeException(e);
132     }
133     catch (NoSuchPaddingException e) {
134         throw new RuntimeException(e);
135     }
136     catch (InvalidKeyException e) {
137         throw new RuntimeException(e);
138     }
139 }
140 }
141
142
143 /**
144  * 入力されたString型二次元配列データを暗号化したものを返す
145  *
146  * @param input String型二次元配列データ
```

```
147  * @return 暗号化された String 型二次元配列データ
148  */
149  public String[][] encrypt (String[][] input) {
150      LogUtil.log(Log.INFO);
151      if (input == null) {
152          LogUtil.log(Log.WARN, "Input data is NULL");
153          return null;
154      }
155
156      String[][] encrypted = new String[input.length][input[0].length];
157
158      try {
159          // 暗号化アルゴリズムに
160             AESを、動作モードにCBCを、パディングにPKCS5を用いる
161          Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
162          cipher.init(Cipher.ENCRYPTMODE, key, iv);
163
164          for (int i = 0; i < input.length; i++) {
165              for (int j = 0; j < input[i].length; j++) {
166                  byte[] result = cipher.doFinal(input[i][j].getBytes());
167                  encrypted[i][j] = Base64.encodeToString(result, Base64.URLSAFE |
168                      Base64.NO_WRAP);
169              }
170          }
171
172          return encrypted;
173      } catch (NoSuchAlgorithmException e) {
174          throw new RuntimeException(e);
175      } catch (NoSuchPaddingException e) {
176          throw new RuntimeException(e);
177      } catch (InvalidKeyException e) {
178          throw new RuntimeException(e);
179      } catch (BadPaddingException e) {
180          throw new RuntimeException(e);
181      } catch (IllegalBlockSizeException e) {
182          throw new RuntimeException(e);
183      }
```

```
186     }
187     catch (InvalidAlgorithmParameterException e) {
188         throw new RuntimeException(e);
189     }
190 }
191
192
193 /**
194  * 入力された暗号化済み String 型二次元配列データを復号したものを返す
195  *
196  * @param input 暗号化された String 型二次元配列データ
197  * @return 復号された String 型二次元配列データ
198  */
199 public String[][] decrypt (String[][] input) {
200     LogUtil.log(Log.INFO);
201     if (input == null) {
202         LogUtil.log(Log.WARN, "Input data is NULL");
203         return null;
204     }
205
206     String[][] decrypted = new String[input.length][input[0].length];
207
208     try {
209         // 復号を行う
210         Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
211         cipher.init(Cipher.DECRYPTMODE, key, iv);
212
213         for (int i = 0; i < input.length; i++) {
214             for (int j = 0; j < input[i].length; j++) {
215                 byte[] result = cipher.doFinal(Base64.decode(input[i][j], Base64.
216                     URLSAFE | Base64.NO_WRAP));
217                 decrypted[i][j] = new String(result);
218                 LogUtil.log(Log.VERBOSE, "Decrypted : " + decrypted[i][j]);
219             }
220         }
221
222         return decrypted;
223     }
224     catch (NoSuchAlgorithmException e) {
225         throw new RuntimeException(e);
226     }
```

```
226     catch (NoSuchPaddingException e) {
227         throw new RuntimeException(e);
228     }
229     catch (InvalidKeyException e) {
230         throw new RuntimeException(e);
231     }
232     catch (BadPaddingException e) {
233         throw new RuntimeException(e);
234     }
235     catch (IllegalBlockSizeException e) {
236         throw new RuntimeException(e);
237     }
238     catch (InvalidAlgorithmParameterException e) {
239         throw new RuntimeException(e);
240     }
241 }
242
243
244 /**
245  * 入力された暗号化済み String 型一次元配列データを復号したものを返す
246  *
247  * @param input 暗号化された String 型一次元配列データ
248  * @return 復号された String 型一次元配列データ
249  */
250 public String[] decrypt (String[] input) {
251     LogUtil.log(Log.INFO);
252     if (input == null) {
253         LogUtil.log(Log.DEBUG, "Input data is NULL");
254         return null;
255     }
256
257     String[] decrypted = new String[input.length];
258
259     try {
260         // 復号を行う
261         Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
262         cipher.init(Cipher.DECRYPT_MODE, key, iv);
263
264         for (int i = 0; i < input.length; i++) {
265             byte[] result = cipher.doFinal(Base64.decode(input[i], Base64.URL_SAFE
                | Base64.NO_WRAP));
```

```
266         decrypted[i] = new String(result);
267     }
268
269     return decrypted;
270 }
271 catch (NoSuchAlgorithmException e) {
272     throw new RuntimeException(e);
273 }
274 catch (NoSuchPaddingException e) {
275     throw new RuntimeException(e);
276 }
277 catch (InvalidKeyException e) {
278     throw new RuntimeException(e);
279 }
280 catch (BadPaddingException e) {
281     throw new RuntimeException(e);
282 }
283 catch (IllegalBlockSizeException e) {
284     throw new RuntimeException(e);
285 }
286 catch (InvalidAlgorithmParameterException e) {
287     throw new RuntimeException(e);
288 }
289 }
290 }
```

C.16 src/com/example/motionauth/Lowpass/Fourier.java

```
1 package com.example.motionauth.Lowpass;
2
3 import android.util.Log;
4 import com.example.motionauth.Authentication.AuthNameInput;
5 import com.example.motionauth.Registration.RegistNameInput;
6 import com.example.motionauth.Utility.LogUtil;
7 import com.example.motionauth.Utility.ManageData;
8 import edu.emory.mathcs.jtransforms.fft.DoubleFFT_1D;
9
10 /**
11  * フーリエ変換を用いたローパスフィルタ
12  * フーリエ変換にはjtransformsライブラリを使用
13  *
```

```
14  * @author Kensuke Kousaka
15  * @see <a href="https://sites.google.com/site/piotrwendykier/software/
      jtransforms">https://sites.google.com/site/piotrwendykier/software/
      jtransforms</a>
16  */
17  public class Fourier {
18      private ManageData mManageData = new ManageData();
19
20      /**
21       * double型三次元配列の入力データに対し,
22       * フーリエ変換を用いてローパスフィルタリングを行ってデータの平滑化を行う
23       *
24       * @param data      データ平滑化を行うdouble型三次元配列データ
25       * @param dataName アウトプット用, データ種別
26       * @return フーリエ変換によるローパスフィルタリングにより滑らかになった
27       *          double型三次元配列データ
28       */
29      public double[][][] LowpassFilter (double[][][] data, String dataName) {
30          LogUtil.log(Log.INFO);
31
32          DoubleFFT_1D realfft = new DoubleFFT_1D(data[0][0].length);
33
34          // フーリエ変換 (ForwardDFT) の実行
35          for (double[][] i : data) {
36              for (double[] j : i) {
37                  realfft.realForward(j);
38              }
39          }
40
41          // 実数部, 虚数部それぞれを入れる配列
42          double[][][] real = new double[data.length][data[0].length][data[0][0].length];
43          double[][][] imaginary = new double[data.length][data[0].length][data[0][0].length];
44
45          int countReal = 0;
46          int countImaginary = 0;
47
48          // 実数部と虚数部に分解
49          for (int i = 0; i < data.length; i++) {
```

```
50     for (int j = 0; j < data[i].length; j++) {
51         for (int k = 0; k < data[i][j].length; k++) {
52             if (k % 2 == 0) {
53                 real[i][j][countReal] = data[i][j][k];
54                 countReal++;
55                 if (countReal == 99) countReal = 0;
56             }
57             else {
58                 imaginary[i][j][countImaginary] = data[i][j][k];
59                 countImaginary++;
60                 if (countImaginary == 99) countImaginary = 0;
61             }
62         }
63     }
64 }
65
66 mManageData.writeDoubleThreeArrayData("ResultFFT", "rFFT" + dataName,
67     RegistNameInput.name, real);
68
69 mManageData.writeDoubleThreeArrayData("ResultFFT", "iFFT" + dataName,
70     RegistNameInput.name, imaginary);
71
72 // パワースペクトルを求めるために，実数部 ( $k$ )，虚数部 ( $k + 1$ ) それぞれを二
73 // 乗して加算し，平方根を取り，絶対値を求める
74
75 double[][][] power = new double[data.length][data[0].length][data[0][0].
76     length / 2];
77
78 for (int i = 0; i < data.length; i++) {
79     for (int j = 0; j < data[i].length; j++) {
80         for (int k = 0; k < data[i][j].length / 2; k++) {
81             power[i][j][k] = Math.sqrt(Math.pow(real[i][j][k], 2) + Math.pow(
82                 imaginary[i][j][k], 2));
83         }
84     }
85 }
```

```
81
82 // ローパスフィルタ処理
83 for (int i = 0; i < data.length; i++) {
84     for (int j = 0; j < data[i].length; j++) {
```



```
85         for (int k = 0; k < data[i][j].length; k++) {
86             if (k > 30) data[i][j][k] = 0;
87         }
88     }
89 }
90
91 for (double[][] i : data) {
92     for (double[] j : i) {
93         realfft.realInverse(j, true);
94     }
95 }
96
97 mManageData.writeDoubleThreeArrayData("AfterFFT", dataName, RegistNameInput
    .name, data);
98
99 return data;
100 }
101
102
103 /**
104  * double型二次元配列の入力データに対し,
105  * フーリエ変換を用いてローパスフィルタリングを行ってデータの平滑化を行う
106  *
107  * @param data データ平滑化を行うdouble型三次元配列データ
108  * @param dataName アウトプット用, データ種別
109  * @return フーリエ変換によるローパスフィルタリングにより滑らかになった
110  *         double型三次元配列データ
111  */
112 public double[][] LowpassFilter (double[][] data, String dataName) {
113     LogUtil.log(Log.INFO);
114
115     DoubleFFT_1D realfft = new DoubleFFT_1D(data[0].length);
116
117     // フーリエ変換 (ForwardDFT) の実行
118     for (double[] i : data) realfft.realForward(i);
119
120     // 実数部, 虚数部それぞれを入れる配列
121     double[][] real = new double[data.length][data[0].length];
122     double[][] imaginary = new double[data.length][data[0].length];
123
124     int countReal = 0;
```

```
124     int countImaginary = 0;
125
126     // 実数部と虚数部に分解
127     for (int i = 0; i < data.length; i++) {
128         for (int j = 0; j < data[i].length; j++) {
129             if (j % 2 == 0) {
130                 real[i][countReal] = data[i][j];
131                 countReal++;
132                 if (countReal == 99) countReal = 0;
133             }
134             else {
135                 imaginary[i][countImaginary] = data[i][j];
136                 countImaginary++;
137                 if (countImaginary == 99) countImaginary = 0;
138             }
139         }
140     }
141 }
142
143 mManageData.writeDoubleTwoArrayData("ResultFFT", "rFFT" + dataName,
    AuthNameInput.name, real);
144 mManageData.writeDoubleTwoArrayData("ResultFFT", "iFFT" + dataName,
    AuthNameInput.name, imaginary);
145
146 // パワースペクトルを求めるために、実数部 ( $k$ )、虚数部 ( $k + 1$ ) それぞれを二
    乗して加算し、平方根を取り、絶対値を求める
147 double[][] power = new double[data.length][data[0].length / 2];
148
149 for (int i = 0; i < data.length; i++) {
150     for (int j = 0; j < data[i].length / 2; j++) {
151         power[i][j] = Math.sqrt(Math.pow(real[i][j], 2) + Math.pow(imaginary[i]
            ][j], 2));
152     }
153 }
154
155 mManageData.writeDoubleTwoArrayData("ResultFFT", "powerFFT" + dataName,
    AuthNameInput.name, power);
156
157 // ローパスフィルタ処理
158 for (int i = 0; i < data.length; i++) {
159     for (int j = 0; j < data[i].length; j++) {
```

```
160         if (j > 30) data[i][j] = 0;
161     }
162 }
163
164 // 逆フーリエ変換 (InverseDFT)
165 for (double[] i : data) realfft.realInverse(i, true);
166
167 mManageData.writeDoubleTwoArrayData("AfterFFT", dataName, AuthNameInput.
    name, data);
168
169 return data;
170 }
171 }
```

C.17 src/com/example/motionauth/Utility/ConvertArrayAndString.java

```
1 package com.example.motionauth.Utility;
2
3 import android.util.Log;
4
5 /**
6  * 文字列型配列データの要素を連結したり，分離して配列に戻すクラス
7  *
8  * @author Kensuke Kousaka
9  */
10 public class ConvertArrayAndString {
11
12     /**
13      * 受け取った配列データを，特定の文字を用いて連結する
14      *
15      * @param input 処理するString型二次元配列データ
16      * @return 連結したString型データ
17      */
18     public String arrayToString (String[][] input) {
19         LogUtil.log(Log.INFO);
20         String join = "", result = "";
21
22         // aaa    bbb    ccc
23         for (String[] i : input) {
24             for (String j : i) {
25                 join += j + ",";
```

```
26     }
27     join += "'";
28 }
29 // a,a,a,'b,b,b','c,c,c,'
30
31 String[] splited = join.split("'");
32 // a,a,a    b,b,b    c,c,c
33
34 for (String i : splited) {
35     if (i.endsWith(",")) {
36         int last = i.lastIndexOf(",");
37         i = i.substring(0, last);
38         // a,a,a
39         result += i + "'";
40     }
41 }
42
43 // a,a,a'b,b,b'c,c,c'
44
45 if (result.endsWith("'")) {
46     int last = result.lastIndexOf("'");
47     result = result.substring(0, last);
48 }
49 // a,a,a'b,b,b'c,c,c
50
51 return result;
52 }
53
54
55 /**
56  * 受け取ったString型データを特定文字列で分割して配列データにする
57  *
58  * @param input 処理するString型データ
59  * @return 分割したString型二次元配列データ
60  */
61 public String[][] stringToArray (String input) {
62     LogUtil.log(Log.INFO);
63     String[] splitDimention = input.split("'");
64     String[][] result = new String[3][100];
65 }
```

```
66     for (int i = 0; i < splitDimention.length; i++) result[i] = splitDimention[
        i].split(",");
67
68     return result;
69 }
70 }
```

C.18 src/com/example/motionauth/Utility/Enum.java

```
1 package com.example.motionauth.Utility;
2
3 /**
4  * 列挙型 . 相関周りの判定の際に使用
5  *
6  * @author Kensuke Kousaka
7  */
8 public class Enum {
9     public static enum MEASURE {
10         BAD, INCORRECT, CORRECT, PERFECT
11     }
12
13     public static enum MODE {
14         MAX, MIN, MEDIAN
15     }
16
17     public static enum TARGET {
18         DISTANCE, ANGLE
19     }
20
21     public final double LOOSE = 0.4;
22     public final double NORMAL = 0.6;
23     public final double STRICT = 0.8;
24 }
```

C.19 src/com/example/motionauth/Utility/LogUtil.java

```
1 package com.example.motionauth.Utility;
2
3 import android.util.Log;
4
5 /**
6  * @author Kensuke Kousaka
```

```
7  */
8  public class LogUtil {
9      private static final String TAG = "Logging";
10
11     private static boolean mIsShowLog = false;
12
13     public static void setShowLog (boolean isShowLog) {
14         mIsShowLog = isShowLog;
15     }
16
17     public static void log () {
18         outputLog(Log.DEBUG, null, null);
19     }
20
21     public static void log (String message) {
22         outputLog(Log.DEBUG, message, null);
23     }
24
25     public static void log (int type) {
26         outputLog(type, null, null);
27     }
28
29     public static void log (int type, String message) {
30         outputLog(type, message, null);
31     }
32
33     public static void log (int type, String message, Throwable throwable) {
34         outputLog(type, message, throwable);
35     }
36
37     private static void outputLog (int type, String message, Throwable throwable)
38     {
39         if (!mIsShowLog) {
40             // ログ出力フラグが立っていない場合は何もしない。
41             return;
42         }
43
44         // ログのメッセージ部分にスタックトレース情報を付加する。
45         if (message == null) {
46             message = getStackTraceInfo();
47         }
48     }
49 }
```

```
47     else {
48         message = getStackTraceInfo() + message;
49     }
50
51     // ログを出力
52     switch (type) {
53         case Log.DEBUG:
54             if (throwable == null) {
55                 Log.d(TAG, message);
56             }
57             else {
58                 Log.d(TAG, message, throwable);
59             }
60             break;
61         case Log.ERROR:
62             if (throwable == null) {
63                 Log.e(TAG, message);
64             }
65             else {
66                 Log.e(TAG, message, throwable);
67             }
68             break;
69         case Log.INFO:
70             if (throwable == null) {
71                 Log.i(TAG, message);
72             }
73             else {
74                 Log.i(TAG, message, throwable);
75             }
76             break;
77         case Log.VERBOSE:
78             if (throwable == null) {
79                 Log.v(TAG, message);
80             }
81             else {
82                 Log.v(TAG, message, throwable);
83             }
84             break;
85         case Log.WARN:
86             if (throwable == null) {
87                 Log.w(TAG, message);
```

```
88     }
89     else {
90         Log.w(TAG, message, throwable);
91     }
92     break;
93 }
94 }
95
96 /**
97  * スタックトレースから呼び出し元の基本情報を取得
98  *
99  * @return <<className#methodName:lineNumber>>
100 */
101 private static String getStackTraceInfo () {
102     // 現在のスタックトレースを取得
103     // 0:VM 1:スレッド 2:getStackTraceInfo() 3:outputLog() 4:logDebug() 等 5:呼
        び出し元
104     StackTraceElement stackTraceElement = Thread.currentThread().getStackTrace
        ()[5];
105
106     String fullName = stackTraceElement.getClassName();
107     String className = fullName.substring(fullName.lastIndexOf(".") + 1);
108     String methodName = stackTraceElement.getMethodName();
109     int lineNumber = stackTraceElement.getLineNumber();
110
111     return "<<" + className + "#" + methodName + ":" + lineNumber + ">> ";
112 }
113 }
```

C.20 src/com/example/motionauth/Utility/ManageData.java

```
1 package com.example.motionauth.Utility;
2
3 import android.content.Context;
4 import android.content.SharedPreferences;
5 import android.os.Environment;
6 import android.util.Log;
7 import com.example.motionauth.Processing.CipherCrypt;
8
9 import java.io.BufferedWriter;
10 import java.io.File;
```



```
11 import java.io.FileOutputStream;
12 import java.io.OutputStreamWriter;
13 import java.util.ArrayList;
14
15
16 /**
17  * データをSDカードに書き込む
18  *
19  * @author Kensuke Kousaka
20  */
21 public class ManageData {
22     /**
23      * Float型の三次元配列データをアウトプットする。
24      * 保存先は, SDカードディレクトリ/folderName/userName/fileName+回数+次元
25      *
26      * @param folderName 保存するフォルダ名
27      * @param dataName 保存するデータ名
28      * @param userName 保存するユーザ名
29      * @param data 保存するfloat型の3次元配列データ
30      */
31     public void writeFloatThreeArrayData (String folderName, String dataName,
32         String userName, float[][][] data) {
33         LogUtil.log(Log.INFO);
34
35         // SDカードのマウント確認
36         String status = Environment.getExternalStorageState();
37
38         // マウントされていない場合
39         if (!status.equals(Environment.MEDIA_MOUNTED)) {
40             LogUtil.log(Log.ERROR, "SDCard not mounted");
41             return;
42         }
43
44         // SDカードのフォルダパスの取得
45         String SD_PATH = Environment.getExternalStorageDirectory().getPath();
46
47         // SDカードにフォルダを作成
48         String FOLDER_PATH = SD_PATH + File.separator + "MotionAuth" + File.
49             separator + folderName + File.separator + userName;
50
51         File file = new File(FOLDER_PATH);
```

```
50
51     try {
52         if (!file.exists()) {
53             // フォルダがない場合
54             if (!file.mkdirs()) {
55                 LogUtil.log(Log.DEBUG, "Make directory error");
56             }
57         }
58     }
59     catch (Exception e) {
60         LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
61     }
62
63     try {
64         // 1ファイルでave_distance_y_1@みたいな感じでやる
65         String dimension = null;
66
67         for (int i = 0; i < data.length; i++) {
68             // X,Y,Zループ
69             for (int j = 0; j < data[i].length; j++) {
70                 if (j == 0) {
71                     dimension = "x";
72                 }
73                 else if (j == 1) {
74                     dimension = "y";
75                 }
76                 else if (j == 2) {
77                     dimension = "z";
78                 }
79
80                 // ファイルパス
81                 String filePath = FOLDERPATH + File.separator + dataName + String.
                        valueOf(i) + dimension;
82                 file = new File(filePath);
83
84                 // ファイルを追記モードで書き込む
85                 FileOutputStream fos = new FileOutputStream(file, false);
86                 OutputStreamWriter osw = new OutputStreamWriter(fos, "UTF-8");
87                 BufferedWriter bw = new BufferedWriter(osw);
88
89                 for (int k = 0; k < data[i][j].length; k++) {
```

```
90         bw.write(dataName + "_" + dimension + "_" + String.valueOf(i + 1) +
91                 "@" + data[i][j][k] + "\n");
92     }
93     bw.close();
94     osw.close();
95     fos.close();
96 }
97 }
98 catch (Exception e) {
99     LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
100 }
101 }
102
103
104 /**
105  * Double型の2次元配列データをアウトプットする .
106  * 保存先は , SDカードディレクトリ /folderName/userName/fileName+次元
107  *
108  * @param folderName 保存するフォルダ名
109  * @param dataName   保存するデータ名
110  * @param userName   保存するユーザ名
111  * @param data       保存するdouble型の2次元配列データ
112  * @return 保存に成功したらtrue , 失敗したらfalseを返す
113  */
114 public boolean writeDoubleTwoArrayData (String folderName, String dataName,
115     String userName, double[][] data) {
116     LogUtil.log(Log.INFO);
117     // SDカードのマウント確認
118     String status = Environment.getExternalStorageState();
119     if (!status.equals(Environment.MEDIA_MOUNTED)) {
120         // マウントされていない場合
121         LogUtil.log(Log.ERROR, "SDCard not mounted");
122         return false;
123     }
124
125     // SDカードのフォルダパスの取得
126     String SD_PATH = Environment.getExternalStorageDirectory().getPath();
127
128     // SDカードにフォルダを作成
```

```
128 String FOLDERPATH = SD_PATH + File.separator + "MotionAuth" + File.
    separator + folderName + File.separator + userName;
129
130 File file = new File(FOLDERPATH);
131
132 try {
133     if (!file.exists()) {
134         // フォルダがない場合
135         if (!file.mkdirs()) {
136             LogUtil.log(Log.ERROR, "Make directory error");
137         }
138     }
139 }
140 catch (Exception e) {
141     LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
142     return false;
143 }
144
145 try {
146     String dimension = null;
147
148     // X,Y,Zループ
149     for (int i = 0; i < data.length; i++) {
150         if (i == 0) {
151             dimension = "x";
152         }
153         else if (i == 1) {
154             dimension = "y";
155         }
156         else if (i == 2) {
157             dimension = "z";
158         }
159
160         // ファイルパス
161         String filePath = FOLDERPATH + File.separator + dataName + String.
            valueOf(i) + dimension;
162         file = new File(filePath);
163
164         // ファイルを追記モードで書き込む
165         FileOutputStream fos = new FileOutputStream(file, false);
166         OutputStreamWriter osw = new OutputStreamWriter(fos, "UTF-8");
```

```
167     BufferedWriter bw = new BufferedWriter(osp);
168
169     for (int j = 0; j < data[i].length; j++) {
170         bw.write(dataName + "_" + dimension + "@" + data[i][j] + "\n");
171     }
172     bw.close();
173     osp.close();
174     fos.close();
175 }
176 }
177 catch (Exception e) {
178     LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
179     return false;
180 }
181 return true;
182 }
183
184
185 /**
186  * Double型の三次元配列データをアウトプットする .
187  * 保存先は , SDカードディレクトリ /folderName/userName/fileName+回数+次元
188  *
189  * @param folderName 保存するフォルダ名
190  * @param dataName   保存するデータ名
191  * @param userName   保存するユーザ名
192  * @param data       保存するdouble型の3次元配列データ
193  */
194 public void writeDoubleThreeArrayData (String folderName, String dataName,
195     String userName, double[][][] data) {
196     LogUtil.log(Log.INFO);
197
198     // SDカードのマウント確認
199     String status = Environment.getExternalStorageState();
200     if (!status.equals(Environment.MEDIA_MOUNTED)) {
201         // マウントされていない場合
202         LogUtil.log(Log.ERROR, "SDCard not mounted");
203     }
204
205     // SDカードのフォルダパスの取得
206     String SD_PATH = Environment.getExternalStorageDirectory().getPath();
```

```
207 // SDカードにフォルダを作成
208 String FOLDER_PATH = SD_PATH + File.separator + "MotionAuth" + File.
    separator + folderName + File.separator + userName;
209
210 File file = new File(FOLDER_PATH);
211
212 try {
213     if (!file.exists()) {
214         // フォルダがない場合
215         if (!file.mkdirs()) {
216             LogUtil.log(Log.ERROR, "Make directory Error");
217         }
218     }
219 }
220 catch (Exception e) {
221     LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
222 }
223
224 try {
225     String dimension = null;
226
227     for (int i = 0; i < data.length; i++) {
228         // X,Y,Zループ
229         for (int j = 0; j < data[i].length; j++) {
230             if (j == 0) {
231                 dimension = "x";
232             }
233             else if (j == 1) {
234                 dimension = "y";
235             }
236             else if (j == 2) {
237                 dimension = "z";
238             }
239
240             // ファイルパス
241             String filePath = FOLDER_PATH + File.separator + dataName + String.
                valueOf(i) + dimension;
242             file = new File(filePath);
243
244             // ファイルを追記モードで書き込む
245             FileOutputStream fos = new FileOutputStream(file, false);
```

```
246     OutputStreamWriter osw = new OutputStreamWriter(fos, "UTF-8");
247     BufferedWriter bw = new BufferedWriter(osw);
248
249     for (int k = 0; k < data[0][0].length; k++) {
250         //bw.write(dataName + "_" + dimension + "_" + String.valueOf(i + 1)
251             + "@" + data[i][j][k] + "\n");
252         bw.write(data[i][j][k] + "\n");
253         bw.flush();
254     }
255     bw.close();
256     osw.close();
257     fos.close();
258 }
259 }
260 catch (Exception e) {
261     LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
262 }
263 }
264
265
266 /**
267  * Double型の2次元配列データをアウトプットする。
268  * 保存先は、SDカードディレクトリ/folderName/userName/fileName+次元
269  *
270  * @param folderName 保存するフォルダ名
271  * @param dataName   保存するデータ名
272  * @param userName   保存するユーザ名
273  * @param data       保存するdouble型の2次元配列データ
274  */
275 public void writeRData (String folderName, String dataName, String userName,
276     double[][] data) {
277     LogUtil.log(Log.INFO);
278
279     // SDカードのマウント確認
280     String status = Environment.getExternalStorageState();
281     if (!status.equals(Environment.MEDIA_MOUNTED)) {
282         // マウントされていない場合
283         LogUtil.log(Log.ERROR, "SDCard not mounted");
284     }
```

```
285 // SDカードのフォルダパスの取得
286 String SD_PATH = Environment.getExternalStorageDirectory().getPath();
287
288 // SDカードにフォルダを作成
289 String FOLDER_PATH = SD_PATH + File.separator + "MotionAuth" + File.
    separator + folderName + File.separator + userName;
290
291 File file = new File(FOLDER_PATH);
292
293 try {
294     if (!file.exists()) {
295         // フォルダがない場合
296         if (!file.mkdirs()) {
297             LogUtil.log(Log.ERROR, "Make directory error");
298         }
299     }
300 }
301 catch (Exception e) {
302     LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
303 }
304
305 try {
306     String dimension = null;
307
308     for (int i = 0; i < data.length; i++) {
309
310         // X,Y,Zループ
311         for (int j = 0; j < data[i].length; j++) {
312             if (j == 0) {
313                 dimension = "x";
314             }
315             else if (j == 1) {
316                 dimension = "y";
317             }
318             else if (j == 2) {
319                 dimension = "z";
320             }
321
322             // ファイルパス
323             String filePath = FOLDER_PATH + File.separator + dataName + String.
                valueOf(i) + dimension;
```



```
324         file = new File(filePath);
325
326         // ファイルを追記モードで書き込む
327         FileOutputStream fos = new FileOutputStream(file, false);
328         OutputStreamWriter osw = new OutputStreamWriter(fos, "UTF-8");
329         BufferedWriter bw = new BufferedWriter(osw);
330
331         bw.write(dataName + "_" + dimension + "_" + String.valueOf(i + 1) + "
            @" + data[i][j] + "\n");
332         bw.close();
333         osw.close();
334         fos.close();
335     }
336 }
337 }
338 catch (Exception e) {
339     LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
340 }
341 }
342
343
344 /**
345  * Double型の2次元配列データをアウトプットする。
346  * 保存先は、SDカードディレクトリ/MotionAuth/folderName/userName
347  *
348  * @param folderName 保存するフォルダ名
349  * @param userName   保存するユーザ名
350  * @param R_accel    保存する1次元double型配列の加速度Rデータ
351  * @param R_gyro     保存する1次元double型配列の角速度Rデータ
352  */
353 public void writeRData (String folderName, String userName, double[] R_accel,
354                        double[] R_gyro) {
355     LogUtil.log(Log.INFO);
356
357     // SDカードのマウント確認
358     String status = Environment.getExternalStorageState();
359     if (!status.equals(Environment.MEDIA_MOUNTED)) {
360         // マウントされていない場合
361         LogUtil.log(Log.ERROR, "SDCard not mounted");
362     }
```

```
363 // SDカードのフォルダパスの取得
364 String SD_PATH = Environment.getExternalStorageDirectory().getPath();
365
366 // SDカードにフォルダを作成
367 String FOLDER_PATH = SD_PATH + File.separator + "MotionAuth" + File.
    separator + folderName;
368
369 File file = new File(FOLDER_PATH);
370
371 try {
372     if (!file.exists()) {
373         // フォルダがない場合
374         if (!file.mkdirs()) {
375             LogUtil.log(Log.ERROR, "Make directory error");
376         }
377     }
378 }
379 catch (Exception e) {
380     LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
381 }
382
383 try {
384     // ファイルパス
385     String filePath = FOLDER_PATH + File.separator + userName;
386     file = new File(filePath);
387
388     // ファイルを追記モードで書き込む
389     FileOutputStream fos = new FileOutputStream(file, false);
390     OutputStreamWriter osw = new OutputStreamWriter(fos, "UTF-8");
391     BufferedWriter bw = new BufferedWriter(osw);
392
393     for (int i = 0; i < 2; i++) {
394         if (i == 0) {
395             for (int j = 0; j < 3; j++) {
396                 if (j == 0) {
397                     bw.write("R_accel_x@" + R_accel[j] + "\n");
398                     bw.flush();
399                 }
400                 if (j == 1) {
401                     bw.write("R_accel_y@" + R_accel[j] + "\n");
402                     bw.flush();
```

```
403     }
404     if (j == 2) {
405         bw.write("R_accel_z@" + R_accel[j] + "\n");
406         bw.flush();
407     }
408 }
409 }
410 else if (i == 1) {
411     for (int j = 0; j < 3; j++) {
412         if (j == 0) {
413             bw.write("R_gyro_x@" + R_gyro[j] + "\n");
414             bw.flush();
415         }
416         if (j == 1) {
417             bw.write("R_gyro_y@" + R_gyro[j] + "\n");
418             bw.flush();
419         }
420         if (j == 2) {
421             bw.write("R_gyro_z@" + R_gyro[j] + "\n");
422             bw.flush();
423         }
424     }
425 }
426 }
427 bw.close();
428 osw.close();
429 fos.close();
430 }
431 catch (Exception e) {
432     LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
433 }
434 }
435
436 // 実験用．新規登録モードにおける登録データをSDカードに保存する
437 public void writeRegisteredDataToSd (String folderName, String userName, double
438     [][] averageDistance, double[][] averageAngle) {
439     LogUtil.log(Log.INFO);
440
441     // SDカードのマウント確認
442     String status = Environment.getExternalStorageState();
443     if (!status.equals(Environment.MEDIA_MOUNTED)) {
```

```
443     LogUtil.log(Log.ERROR, "SDCard not mounted");
444 }
445
446 String SD_PATH = Environment.getExternalStorageDirectory().getPath();
447
448 String FOLDER_PATH = SD_PATH + File.separator + "MotionAuth" + File.
    separator + folderName;
449
450 File file = new File(FOLDER_PATH);
451
452 try {
453     if (!file.exists()) {
454         if (!file.mkdirs()) {
455             LogUtil.log(Log.ERROR, "Make directory error");
456         }
457     }
458 }
459 catch (Exception e) {
460     LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
461 }
462
463 try {
464     String filePath = FOLDER_PATH + File.separator + userName;
465     file = new File(filePath);
466
467     FileOutputStream fileOutputStream = new FileOutputStream(file, false);
468     OutputStreamWriter outputStreamWriter = new OutputStreamWriter(
        fileOutputStream, "UTF-8");
469     BufferedWriter bw = new BufferedWriter(outputStreamWriter);
470
471     // 距離データ書き込み
472     for (int i = 0; i < averageDistance.length; i++) {
473         for (int j = 0; j < averageDistance[i].length; j++) {
474             switch (i) {
475                 case 0:
476                     // dimension x
477                     bw.write(String.valueOf(averageDistance[i][j]) + "\n");
478                     break;
479                 case 1:
480                     // dimension y
481                     bw.write(String.valueOf(averageDistance[i][j]) + "\n");
```

```
482         break;
483     case 2:
484         // dimension z
485         bw.write(String.valueOf(averageDistance[i][j]) + "\n");
486         break;
487     }
488 }
489 }
490
491 // 角度データ書き込み
492 for (int i = 0; i < averageAngle.length; i++) {
493     for (int j = 0; j < averageAngle[i].length; j++) {
494         switch (i) {
495             case 0:
496                 // dimension x
497                 bw.write(String.valueOf(averageAngle[i][j]) + "\n");
498                 break;
499             case 1:
500                 // dimension y
501                 bw.write(String.valueOf(averageAngle[i][j]) + "\n");
502                 break;
503             case 2:
504                 // dimension z
505                 bw.write(String.valueOf(averageAngle[i][j]) + "\n");
506                 break;
507         }
508     }
509 }
510 }
511
512 bw.close();
513 outputStreamWriter.close();
514 fileOutputStream.close();
515 }
516 catch (Exception e) {
517     LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
518 }
519 }
520
521
522 public void writeRpoint (String folderName, String userName, double data) {
```

```
523     LogUtil.log(Log.INFO);
524
525     // SDカードのマウント確認
526     String status = Environment.getExternalStorageState();
527     if (!status.equals(Environment.MEDIA_MOUNTED)) {
528         LogUtil.log(Log.ERROR, "SDCard not mounted");
529     }
530
531     String SD_PATH = Environment.getExternalStorageDirectory().getPath();
532
533     String FOLDER_PATH = SD_PATH + File.separator + "MotionAuth" + File.
        separator + folderName;
534
535     File file = new File(FOLDER_PATH);
536
537     try {
538         if (!file.exists()) {
539             if (!file.mkdirs()) {
540                 LogUtil.log(Log.ERROR, "Make directory error");
541             }
542         }
543     }
544     catch (Exception e) {
545         LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
546     }
547
548     try {
549         String filePath = FOLDER_PATH + File.separator + userName;
550         file = new File(filePath);
551
552         FileOutputStream fileOutputStream = new FileOutputStream(file, false);
553         OutputStreamWriter outputStreamWriter = new OutputStreamWriter(
            fileOutputStream);
554         BufferedWriter bufferedWriter = new BufferedWriter(outputStreamWriter);
555
556         bufferedWriter.write(String.valueOf(data));
557
558         bufferedWriter.close();
559         outputStreamWriter.close();
560         fileOutputStream.close();
561     }
```

```
562     catch (Exception e) {
563         LogUtil.log(Log.ERROR, e.getMessage(), e.getCause());
564     }
565 }
566
567
568 /**
569  * RegistMotionより渡された、認証のキーとなるデータをアウトプットする
570  *
571  * @param userName      保存するユーザ名
572  * @param averageDistance 保存する距離データ
573  * @param averageAngle   保存する角度データ
574  * @param ampValue       データ増幅値
575  * @param context        呼び出し元のコンテキスト
576  */
577 // 受け取ったデータをCipherクラスに渡し、暗号化されたデータを保存する
578 public void writeRegisteredData (String userName, double[][] averageDistance,
579     double[][] averageAngle, double ampValue, Context context) {
580
581     LogUtil.log(Log.INFO);
582
583     // 暗号処理を担うオブジェクトを生成
584     CipherCrypt mCipherCrypt = new CipherCrypt(context);
585
586     String[][] averageDistanceStr = new String[averageDistance.length][
587         averageDistance[0].length];
588     String[][] averageAngleStr = new String[averageAngle.length][averageAngle
589         [0].length];
590
591     // 暗号化処理
592     // double型二次元配列で受け取ったデータをString型二次元配列に変換する
593     for (int i = 0; i < averageDistance.length; i++) {
594         for (int j = 0; j < averageDistance[i].length; j++) {
595             averageDistanceStr[i][j] = String.valueOf(averageDistance[i][j]);
596             averageAngleStr[i][j] = String.valueOf(averageAngle[i][j]);
597         }
598     }
599
600     // 暗号化
601     String[][] encryptedAvarageDistanceStr = mCipherCrypt.encrypt(
602         averageDistanceStr);
```

```
599     String[][] encryptedAverageAngleStr = mCipherCrypt.encrypt(averageAngleStr
600         );
601     // 配列データを特定文字列を挟んで連結する
602     ConvertArrayAndString mConvertArrayAndString = new ConvertArrayAndString();
603     String registDistanceData = mConvertArrayAndString.arrayToString(
604         encryptedAvarageDistanceStr);
605     String registAngleData = mConvertArrayAndString.arrayToString(
606         encryptedAverageAngleStr);
607     Context mContext = context.getApplicationContext();
608     SharedPreferences userPref = mContext.getSharedPreferences("UserList",
609         Context.MODE_PRIVATE);
610     SharedPreferences.Editor userPrefEditor = userPref.edit();
611     userPrefEditor.putString(userName, "");
612     userPrefEditor.apply();
613     SharedPreferences preferences = mContext.getSharedPreferences("MotionAuth",
614         Context.MODE_PRIVATE);
615     SharedPreferences.Editor editor = preferences.edit();
616     editor.putString(userName + "distance", registDistanceData);
617     editor.putString(userName + "angle", registAngleData);
618     editor.putString(userName + "amplify", String.valueOf(ampValue));
619     editor.apply();
620 }
621
622
623 /**
624  * SharedPreferencesに保存されたデータを読み取るクラス
625  *
626  * @param context アプリケーション固有のプリファレンスを取得する際に必要とな
627     るコンテキスト
628  * @param userName 読み取るユーザ名
629  * @return 読み取ったdouble型二次元配列データ
630  */
631 public ArrayList<double[][]> readRegisteredData (Context context, String
632     userName) {
633     LogUtil.log(Log.INFO);
634     Context mContext = context.getApplicationContext();
```



```
633
634     SharedPreferences preferences = mContext.getSharedPreferences("MotionAuth",
        Context.MODE_PRIVATE);
635
636     String registeredDistanceData = preferences.getString(userName + "distance",
        "");
637     String registeredAngleData = preferences.getString(userName + "angle", "");
638
639     if ("".equals(registeredDistanceData)) throw new RuntimeException();
640
641     ConvertArrayAndString mConvertArrayAndString = new ConvertArrayAndString();
642     CipherCrypt mCipherCrypt = new CipherCrypt(context);
643
644     String[][] decryptedDistance = mCipherCrypt.decrypt(mConvertArrayAndString.
        stringToArray(registeredDistanceData));
645     String[][] decryptedAngle = mCipherCrypt.decrypt(mConvertArrayAndString.
        stringToArray(registeredAngleData));
646
647     double[][] distance = new double[3][100], angle = new double[3][100];
648
649     for (int i = 0; i < decryptedDistance.length; i++) {
650         for (int j = 0; j < decryptedDistance[i].length; j++) {
651             distance[i][j] = Double.valueOf(decryptedDistance[i][j]);
652             angle[i][j] = Double.valueOf(decryptedAngle[i][j]);
653         }
654     }
655
656     ArrayList<double[][]> result = new ArrayList<>();
657     result.add(distance);
658     result.add(angle);
659
660     return result;
661 }
662 }
```

C.21 res/layout/activity_start.xml

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
```

```
5      android:paddingBottom="@dimen/activity_vertical_margin"
6      android:paddingLeft="@dimen/activity_horizontal_margin"
7      android:paddingRight="@dimen/activity_horizontal_margin"
8      android:paddingTop="@dimen/activity_vertical_margin"
9      tools:context=".Start">
10
11      <!-- タイトル -->
12      <TextView
13          android:id="@+id/title"
14          android:layout_width="wrap_content"
15          android:layout_height="wrap_content"
16          android:layout_alignParentTop="true"
17          android:layout_centerHorizontal="true"
18          android:layout_marginTop="55dp"
19          android:text="@string/app_name"
20          android:textAppearance="?android:attr/textAppearanceMedium" />
21
22      <!-- スタートボタン -->
23      <Button
24          android:id="@+id/start"
25          android:layout_width="wrap_content"
26          android:layout_height="wrap_content"
27          android:layout_alignParentBottom="true"
28          android:layout_centerHorizontal="true"
29          android:layout_marginBottom="80dp"
30          android:text="@string/start" />
31
32 </RelativeLayout>
```

C.22 res/layout/activity_regist_name_input.xml

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:paddingBottom="@dimen/activity_vertical_margin"
6     android:paddingLeft="@dimen/activity_horizontal_margin"
7     android:paddingRight="@dimen/activity_horizontal_margin"
8     android:paddingTop="@dimen/activity_vertical_margin"
9     tools:context=".RegistNameInput">
10
```

```
11 <TextView
12     android:id="@+id/nameInputTextView"
13     android:layout_width="wrap_content"
14     android:layout_height="wrap_content"
15     android:layout_alignParentTop="true"
16     android:layout_centerHorizontal="true"
17     android:layout_marginTop="85dp"
18     android:text="@string/name_input"
19     android:textAppearance="?android:attr/textAppearanceMedium" />
20
21 <EditText
22     android:id="@+id/nameInputEditText"
23     android:layout_width="wrap_content"
24     android:layout_height="wrap_content"
25     android:layout_below="@+id/nameInputTextView"
26     android:layout_centerHorizontal="true"
27     android:layout_marginTop="49dp"
28     android:ems="10"
29     android:inputType="textPersonName">
30
31     <requestFocus />
32 </EditText>
33
34 <Button
35     android:id="@+id/okButton"
36     android:layout_width="wrap_content"
37     android:layout_height="wrap_content"
38     android:layout_alignParentBottom="true"
39     android:layout_centerHorizontal="true"
40     android:layout_marginBottom="85dp"
41     android:text="@string/OK" />
42
43 </RelativeLayout>
```

C.23 res/layout/activity_regist_motion.xml

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:paddingBottom="@dimen/activity_vertical_margin"
```

```
6      android:paddingLeft="@dimen/activity_horizontal_margin"
7      android:paddingRight="@dimen/activity_horizontal_margin"
8      android:paddingTop="@dimen/activity_vertical_margin"
9      tools:context=".RegistMotion">
10
11      <TextView
12          android:id="@+id/textView1"
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:layout_alignParentBottom="true"
16          android:layout_marginBottom="122dp"
17          android:text="@string/ato"
18          android:textAppearance="?android:attr/textAppearanceMedium" />
19
20      <TextView
21          android:id="@+id/secondTextView"
22          android:layout_width="wrap_content"
23          android:layout_height="wrap_content"
24          android:layout_alignTop="@+id/textView1"
25          android:layout_centerHorizontal="true"
26          android:layout_marginTop="31dp"
27          android:text="@string/third"
28          android:textAppearance="?android:attr/textAppearanceLarge" />
29
30      <Button
31          android:id="@+id/button1"
32          android:layout_width="wrap_content"
33          android:layout_height="wrap_content"
34          android:layout_below="@+id/textView4"
35          android:layout_centerHorizontal="true"
36          android:text="@string/getData" />
37
38      <TextView
39          android:id="@+id/textView2"
40          android:layout_width="wrap_content"
41          android:layout_height="wrap_content"
42          android:layout_alignParentTop="true"
43          android:layout_centerHorizontal="true"
44          android:text="@string/hoge"
45          android:textAppearance="?android:attr/textAppearanceLarge" />
46
```

```
47 <TextView
48     android:id="@+id/textView4"
49     android:layout_width="wrap_content"
50     android:layout_height="wrap_content"
51     android:layout_alignParentRight="true"
52     android:layout_below="@+id/secondTextView"
53     android:layout_marginRight="16dp"
54     android:layout_marginTop="22dp"
55     android:text="@string/count"
56     android:textAppearance="?android:attr/textAppearanceMedium" />
57
58 <LinearLayout
59     android:layout_width="wrap_content"
60     android:layout_height="wrap_content"
61     android:layout_above="@+id/textView1"
62     android:layout_alignLeft="@+id/textView1"
63     android:layout_alignParentRight="true"
64     android:layout_below="@+id/textView2"
65     android:layout_marginBottom="10dp"
66     android:layout_marginTop="10dp"
67     android:orientation="vertical">
68
69     <LinearLayout
70         android:layout_width="match_parent"
71         android:layout_height="wrap_content">
72
73         <TextView
74             android:id="@+id/textView3"
75             android:layout_width="wrap_content"
76             android:layout_height="wrap_content"
77             android:text="@string/one"
78             android:textAppearance="?android:attr/textAppearanceMedium" />
79
80         <TextView
81             android:id="@+id/textView5"
82             android:layout_width="wrap_content"
83             android:layout_height="wrap_content"
84             android:text="@string/description1"
85             android:textAppearance="?android:attr/textAppearanceMedium" />
86     </LinearLayout>
87
```

```
88     <LinearLayout
89         android:layout_width="match_parent"
90         android:layout_height="wrap_content">
91
92         <TextView
93             android:id="@+id/textView6"
94             android:layout_width="wrap_content"
95             android:layout_height="wrap_content"
96             android:text="@string/two"
97             android:textAppearance="?android:attr/textAppearanceMedium" />
98
99         <TextView
100             android:id="@+id/textView7"
101             android:layout_width="wrap_content"
102             android:layout_height="wrap_content"
103             android:text="@string/regist_description2"
104             android:textAppearance="?android:attr/textAppearanceMedium" />
105     </LinearLayout>
106 </LinearLayout>
107
108 </RelativeLayout>
```

C.24 res/layout/activity_auth_name_input.xml

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:paddingBottom="@dimen/activity_vertical_margin"
6     android:paddingLeft="@dimen/activity_horizontal_margin"
7     android:paddingRight="@dimen/activity_horizontal_margin"
8     android:paddingTop="@dimen/activity_vertical_margin"
9     tools:context=".RegistNameInput">
10
11     <TextView
12         android:id="@+id/nameInputTextView"
13         android:layout_width="wrap_content"
14         android:layout_height="wrap_content"
15         android:layout_alignParentTop="true"
16         android:layout_centerHorizontal="true"
17         android:layout_marginTop="85dp"
```

```
18         android:text="@string/name_input"
19         android:textAppearance="?android:attr/textAppearanceMedium" />
20
21     <EditText
22         android:id="@+id/nameInputEditText"
23         android:layout_width="wrap_content"
24         android:layout_height="wrap_content"
25         android:layout_below="@+id/nameInputTextView"
26         android:layout_centerHorizontal="true"
27         android:layout_marginTop="49dp"
28         android:ems="10"
29         android:inputType="textPersonName">
30
31         <requestFocus />
32     </EditText>
33
34     <Button
35         android:id="@+id/okButton"
36         android:layout_width="wrap_content"
37         android:layout_height="wrap_content"
38         android:layout_alignParentBottom="true"
39         android:layout_centerHorizontal="true"
40         android:layout_marginBottom="85dp"
41         android:text="@string/OK" />
42
43 </RelativeLayout>
```

C.25 res/layout/activity_auth_motion.xml

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:paddingBottom="@dimen/activity_vertical_margin"
6     android:paddingLeft="@dimen/activity_horizontal_margin"
7     android:paddingRight="@dimen/activity_horizontal_margin"
8     android:paddingTop="@dimen/activity_vertical_margin"
9     tools:context=".AuthMotion">
10
11     <TextView
12         android:id="@+id/textView1"
```

```
13         android:layout_width="wrap_content"
14         android:layout_height="wrap_content"
15         android:layout_alignParentTop="true"
16         android:layout_centerHorizontal="true"
17         android:text="@string/hoge"
18         android:textAppearance="?android:attr/textAppearanceLarge" />
19
20     <Button
21         android:id="@+id/button1"
22         android:layout_width="wrap_content"
23         android:layout_height="wrap_content"
24         android:layout_alignParentBottom="true"
25         android:layout_centerHorizontal="true"
26         android:text="@string/getData" />
27
28     <TextView
29         android:id="@+id/secondTextView"
30         android:layout_width="wrap_content"
31         android:layout_height="wrap_content"
32         android:layout_above="@+id/button1"
33         android:layout_centerHorizontal="true"
34         android:layout_marginBottom="29dp"
35         android:text="@string/first"
36         android:textAppearance="?android:attr/textAppearanceLarge" />
37
38     <TextView
39         android:id="@+id/textView4"
40         android:layout_width="wrap_content"
41         android:layout_height="wrap_content"
42         android:layout_above="@+id/button1"
43         android:layout_alignParentRight="true"
44         android:layout_marginBottom="11dp"
45         android:text="@string/count"
46         android:textAppearance="?android:attr/textAppearanceMedium" />
47
48     <TextView
49         android:id="@+id/textView3"
50         android:layout_width="wrap_content"
51         android:layout_height="wrap_content"
52         android:layout_above="@+id/secondTextView"
53         android:layout_alignParentLeft="true"
```



```
54         android:text="@string/ato"
55         android:textAppearance="?android:attr/textAppearanceMedium" />
56
57     <LinearLayout
58         android:layout_width="wrap_content"
59         android:layout_height="wrap_content"
60         android:layout_above="@+id/textView3"
61         android:layout_alignLeft="@+id/textView3"
62         android:layout_alignParentRight="true"
63         android:layout_below="@+id/textView1"
64         android:layout_marginBottom="10dp"
65         android:layout_marginTop="10dp"
66         android:orientation="vertical">
67
68         <LinearLayout
69             android:layout_width="match_parent"
70             android:layout_height="wrap_content">
71
72             <TextView
73                 android:id="@+id/textView2"
74                 android:layout_width="wrap_content"
75                 android:layout_height="wrap_content"
76                 android:text="@string/one"
77                 android:textAppearance="?android:attr/textAppearanceMedium" />
78
79             <TextView
80                 android:id="@+id/textView5"
81                 android:layout_width="wrap_content"
82                 android:layout_height="wrap_content"
83                 android:text="@string/description1"
84                 android:textAppearance="?android:attr/textAppearanceMedium" />
85         </LinearLayout>
86
87         <LinearLayout
88             android:layout_width="match_parent"
89             android:layout_height="wrap_content">
90
91             <TextView
92                 android:id="@+id/textView6"
93                 android:layout_width="wrap_content"
94                 android:layout_height="wrap_content"
```

```
95         android:text="@string/two"
96         android:textAppearance="?android:attr/textAppearanceMedium" />
97
98     <TextView
99         android:id="@+id/textView7"
100        android:layout_width="wrap_content"
101        android:layout_height="wrap_content"
102        android:text="@string/auth_description2"
103        android:textAppearance="?android:attr/textAppearanceMedium" />
104    </LinearLayout>
105
106    <LinearLayout
107        android:layout_width="match_parent"
108        android:layout_height="wrap_content">
109
110        <TextView
111            android:id="@+id/textView8"
112            android:layout_width="wrap_content"
113            android:layout_height="wrap_content"
114            android:text="@string/three"
115            android:textAppearance="?android:attr/textAppearanceMedium" />
116
117        <TextView
118            android:id="@+id/textView9"
119            android:layout_width="wrap_content"
120            android:layout_height="wrap_content"
121            android:text="@string/description3"
122            android:textAppearance="?android:attr/textAppearanceMedium" />
123    </LinearLayout>
124 </LinearLayout>
125
126 </RelativeLayout>
```

C.26 res/layout/activity_registrant_list.xml

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent"
5   android:paddingBottom="@dimen/activity_vertical_margin"
6   android:paddingLeft="@dimen/activity_horizontal_margin"
```

```
7      android:paddingRight="@dimen/activity_horizontal_margin"
8      android:paddingTop="@dimen/activity_vertical_margin"
9      tools:context=".RegistrantList">
10
11      <ListView
12          android:id="@+id/listView1"
13          android:layout_width="match_parent"
14          android:layout_height="wrap_content"
15          android:layout_alignParentLeft="true"
16          android:layout_alignParentTop="true"></ListView>
17
18 </RelativeLayout>
```

C.27 res/layout/activity_view_registered_data.xml

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:paddingBottom="@dimen/activity_vertical_margin"
6     android:paddingLeft="@dimen/activity_horizontal_margin"
7     android:paddingRight="@dimen/activity_horizontal_margin"
8     android:paddingTop="@dimen/activity_vertical_margin"
9     tools:context=".ViewRegisteredData">
10
11     <ListView
12         android:id="@+id/listView1"
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:layout_alignParentLeft="true"
16         android:layout_alignParentTop="true"></ListView>
17
18 </RelativeLayout>
```

C.28 res/layout/activity_view_registered_rdata.xml

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:paddingBottom="@dimen/activity_vertical_margin"
6     android:paddingLeft="@dimen/activity_horizontal_margin"
```

```
7      android:paddingRight="@dimen/activity_horizontal_margin"
8      android:paddingTop="@dimen/activity_vertical_margin"
9      tools:context=".ViewRegisteredData">
10
11      <ListView
12          android:id="@+id/listView1"
13          android:layout_width="match_parent"
14          android:layout_height="wrap_content"
15          android:layout_alignParentLeft="true"
16          android:layout_alignParentTop="true"></ListView>
17
18 </RelativeLayout>
```

C.29 res/layout/activity_view_auth_rdata.xml

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:paddingBottom="@dimen/activity_vertical_margin"
6     android:paddingLeft="@dimen/activity_horizontal_margin"
7     android:paddingRight="@dimen/activity_horizontal_margin"
8     android:paddingTop="@dimen/activity_vertical_margin"
9     tools:context=".ViewRegisteredData">
10
11     <ListView
12         android:id="@+id/listView1"
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:layout_alignParentLeft="true"
16         android:layout_alignParentTop="true"></ListView>
17
18 </RelativeLayout>
```

C.30 res/layout/seekdialog.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2
3 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/dialog_root"
5     android:orientation="vertical"
6     android:layout_width="wrap_content"
```

```
7      android:layout_height="wrap_content">
8
9      <TextView
10          android:id="@+id/thresholdtext"
11          android:layout_width="fill_parent"
12          android:layout_height="wrap_content"
13          android:layout_gravity="center"
14          android:gravity="center"
15          android:text="@string/text"></TextView>
16
17      <SeekBar
18          android:id="@+id/threshold"
19          android:layout_width="fill_parent"
20          android:layout_height="wrap_content">
21
22      </SeekBar>
23      <TextView
24          android:layout_width="fill_parent"
25          android:layout_height="wrap_content"
26          android:layout_gravity="center"
27          android:gravity="center"
28          android:id="@+id/ampvaltext"
29          android:text="@string/text"/>
30      <SeekBar
31          android:layout_width="match_parent"
32          android:layout_height="wrap_content"
33          android:id="@+id/ampval"/>
34
35 </LinearLayout>
```

C.31 res/menu/regist_motion.xml

```
1 <menu xmlns:android="http://schemas.android.com/apk/res/android">
2
3     <!--<item-->
4     <!--android:id="@+id/action_settings"-->
5     <!--android:orderInCategory="100"-->
6     <!--android:showAsAction="never"-->
7     <!--android:title="@string/action_settings"/>-->
8     <item
9         android:id="@+id/change_range_value"
```

```

10         android:title="@string/amplifier_settings"/>
11     <item
12         android:id="@+id/reset"
13         android:title="@string/reset" />
14 </menu>

```

C.32 res/values/configs.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <bool name="isShowLog">true</bool>
4 </resources>

```

C.33 res/values/strings.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3
4     <string name="app_name">MotionAuth</string>
5     <string name="action_settings">Settings</string>
6     <string name="start">Start</string>
7     <string name="title_activity_regist_name_input">RegistNameInput</string>
8     <string name="name_input">あなたの名前を入力してください</string>
9     <string name="OK">OK</string>
10    <string name="title_activity_regist_motion">RegistMotion</string>
11    <string name="please_read">さん読んでね!! </string>
12    <string name="one">^e2^91^a0</string>
13    <string name="description1">モーションデータ取得ボタンを押すと、3秒間の準備時間の後、ス
        マートフォンの動きを取得し始めます。3秒間でお好きな動きをどうぞ</string>
14    <string name="two">^e2^91^a1</string>
15    <string name="regist_description2">データ取得
        は3回行います。データ取得が正常に完了すれば、登録は終了です。</string>
16    <string name="auth_description2">データ取得は1回行います。</string>
17    <string name="three">^e2^91^a2</string>
18    <string name="description3">データ取得完了後、認証を行います。</string>
19    <string name="ato">あと</string>
20    <string name="count">回</string>
21    <string name="third">3</string>
22    <string name="first">1</string>
23    <string name="hoge">hoge</string>
24    <string name="getData">モーションデータ取得</string>
25    <string name="title_activity_auth_name_input">AuthNameInput</string>

```

```

26     <string name="title_activity_auth_motion">AuthMotion</string>
27     <string name="title_activity_registrant_list">RegistrantList</string>
28     <string name="title_activity_view_registered_data">ViewRegisteredData</string>
29     <string name="amplifier_settings">増幅器設定</string>
30     <string name="title_activity_view_registered_rdata">ViewRegisteredRData</string
31         >
32     <string name="reset">リセット</string>
33     <string name="title_activity_view_auth_rdata">ViewAuthRData</string>
34     <string name="text">text</string>
35 </resources>

```

C.34 res/values/styles.xml

```

1 <resources>
2
3     <!--
4         Base application theme, dependent on API level. This theme is replaced
5         by AppBaseTheme from res/values-vXX/styles.xml on newer devices.
6     -->
7     <style name="AppBaseTheme" parent="android:Theme.Black">
8         <!--
9             Theme customizations available in newer API levels can go in
10            res/values-vXX/styles.xml, while customizations related to
11            backward-compatibility can go here.
12        -->
13    </style>
14
15    <!-- Application theme. -->
16    <style name="AppTheme" parent="AppBaseTheme">
17        <!-- All customizations that are NOT specific to a particular API-level
18            can go here. -->
19    </style>
20 </resources>

```

C.35 AndroidManifest.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest
3     package="com.example.motionauth"
4     xmlns:android="http://schemas.android.com/apk/res/android"

```

```
5  android:versionCode="1"
6  android:versionName="1.0">
7
8  <!-- 対象となるAndroid Versionの指定 -->
9  <uses-sdk
10     android:minSdkVersion="9"
11     android:targetSdkVersion="17" />
12
13  <!-- SDカードへの書き込み権限 -->
14  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" /
15     >
16
17  <uses-permission android:name="android.permission.VIBRATE" />
18
19  <application
20     android:allowBackup="true"
21     android:debuggable="true"
22     android:icon="@drawable/ic_launcher"
23     android:label="@string/app_name"
24     android:theme="@style/AppTheme">
25     <activity
26         android:name="com.example.motionauth.Start"
27         android:label="@string/app_name"
28         android:screenOrientation="portrait">
29         <intent-filter>
30             <action android:name="android.intent.action.MAIN" />
31
32             <category android:name="android.intent.category.LAUNCHER" />
33         </intent-filter>
34     </activity>
35     <activity
36         android:name=".Registration.RegistNameInput"
37         android:label="@string/title_activity_regist_name_input"
38         android:screenOrientation="portrait"></activity>
39     <activity
40         android:name=".Registration.RegistMotion"
41         android:label="@string/title_activity_regist_motion"
42         android:screenOrientation="portrait"></activity>
43     <activity
44         android:name=".Authentication.AuthNameInput"
45         android:label="@string/title_activity_auth_name_input"
46         android:screenOrientation="portrait"></activity>
```



```
45     <activity
46         android:name=".Authentication.AuthMotion"
47         android:label="@string/title_activity_auth_motion"
48         android:screenOrientation="portrait"></activity>
49     <activity
50         android:name=".ViewDataList.RegistrantList"
51         android:label="@string/title_activity_registrant_list"
52         android:screenOrientation="portrait"></activity>
53     <activity
54         android:name=".ViewDataList.ViewRegisteredData"
55         android:label="@string/title_activity_view_registered_data"
56         android:screenOrientation="portrait"></activity>
57     <activity
58         android:name=".ViewDataList.ViewRegisteredRData"
59         android:label="@string/title_activity_view_registered_rdata"
60         android:screenOrientation="portrait"></activity>
61     <activity
62         android:name=".ViewDataList.ViewAuthRData"
63         android:label="@string/title_activity_view_auth_rdata"
64         android:screenOrientation="portrait"></activity>
65 </application>
66 </manifest>
```