平成 29 年度　特別研究報告書

一筆書きを用いた個人識別による スマートフォンのロック解除手法の検討

龍谷大学　理工学部　情報メディア学科

T140479　中川翔平

指導教員　三好　力　教授

内容梗概

現代社会において、暗証番号やパスワードの不正利用による個人情報の流出やなりすましが大きな問題となっている。特に、多くの人々が利用しているスマートフォンは悪用されると持ち主だけでなく、電話帳や SNS に登録されている他人にも影響、被害が出てしまう。しかし、現在スマートフォンやタブレット端末の画面ロックには主にタッチスクリーンを用いた PIN コードと呼ばれる 4 桁の数字のパスワードやパターンと呼ばれる 9 つの点から 4 つ以上の点を選び一筆書きで結ぶ方法が用いられており、それらは持ち主から推測できてしまうパスワードやロック解除中に画面を覗かれてしまうだけで解除される危険性がある。そのため、持ち主から推測されにくく覗かれても持ち主以外がロック解除できないような方法が必要である。本研究では新しい画面ロック解除方法として、一筆書きを描き、一定時間ごとのタッチしている位置を測定することにより個人を識別する方法を提案している。そして、同方法は持ち主の解除成功率、他人の解除失敗率ともに高く、また、個人差が少ないことを確認し、タッチスクリーンを用いた画面ロック解除方法として利用できることを示している。

目次

1　はじめに

1.1　研究背景

近年、日本でのインターネット利用者は増え続け、総務省の情報通信白書 [1)]によると 2015 年末時点でのインターネット利用者数は前年より 28 万人増加し 1 億 46 万人となり、人口普及率は 83.0%になっている。特に、スマートフォンやタブレット端末は近年、利用者数が増加しており、スマートフォンはインターネット利用者の半分以上が利用する端末となっている。また、現代社会において、暗証番号やパスワードの不正利用による個人情報の流出やなりすましが大きな問題となっている。スマートフォンを悪用され起こる被害として電話帳に登録されている人の電話番号やメールアドレスの流出、ネットショッピングなどでの電子マネー、クレジットカードの不正使用、SNS でのなりすましや乗っ取りなど持ち主だけでなく、電話帳や SNS に登録されている人にも影響、被害が出てしまうものがある。しかし、現在スマートフォンやタブレット端末の画面ロックには主にタッチスクリーンを用いた PIN コードと呼ばれる 4 桁の数字のパスワードやパターンと呼ばれる 9 つの点から 4 つ以上の点を選び一筆書きで結ぶ方法が用いられているが、それらは持ち主から推測できてしまうパスワードやロック解除中に画面を覗かれてしまうだけで誰にでもロック解除ができてしまうものである。そのため、持ち主から推測されにくく覗かれても持ち主以外がロック解除できないような方法が必要である。

1.2　既存技術

1.2.1 スワイプ

画面下方に表示されている錠マークを画面上方にスワイプしロックを解除する方法で、解除方法が簡単なため誰にでも解除できるため、買ったばかりのスマートフォンやタブレット端末ではこのロック解除が初期設定として採用されていることが多い。しかし、セキュリティ的にはほとんど意味がなく、なにかの拍子に電源ボタンが押された時などに誤操作、誤作動を防ぐ効果しかない。

1.2.2 PIN コード

4桁以上の数字を組み合わせたコードを入力しロックを解除する方法で、語呂合わせや生年月日などの覚えやすい数字を設定できるが、持ち主の個人情報から推測されてしまう危険性やロック解除中に画面を覗かれてしまうと簡単に覚えられてしまう危険性がある。

1.2.3 パスワード

4 桁以上の英数字、記号を組み合わせたコードを入力しロックを解除する方法で、PIN コードよりも使用できる文字が増え、他人に推測されにくいものの PIN コードより入力が面倒になってしまいがちである。また、ロック解除中に画面を覗かれてしまうと簡単に覚えられてしまう危険性は解決できていない。

1.2.4 パターン

画面に表示されている 9 つの点から 4 つ以上の点を通るようにスワイプすることでロックを解除する方法で、ロック解除中に画面を覗かれた場合、簡単なパターンでは、他人に覚えられてしまう恐れがあるが、複雑なパターンを設定すれば、覚えられてしまう危険性は低い。しかし、複雑なパターンは覚えにくく、忘れてしまう可能性がある。

1.2.5 指紋認証・顔認証

指紋認証は、指紋の情報を登録し、それをセンサで読み取ることでロックを解除する方法で、個人特有の指紋を利用しているため、セキュリティ面は他のロック解除方法よりも高い。しかし、汗や汚れなどちょっとした原因でロック解除できなくなってしまう場合がある。また、現状では指紋認証がうまくいかなかった時のために、PIN コード、パスワード、パターンなどを登録する必要があり、指紋認証だけで利用することができない。顔認証は、顔の画像を登録し、顔をカメラに移すことでロックを解除する方法で、顔の画像を利用しているため、セキュリティ面は他のロック解除方法よりも高い。しかし、カメラを用いるため、認証率は周りの明るさに影響されてしまう。また、指紋認証と同じく顔認証だけで利用することができない。

1.3 提案手法

既存のロック解除方法は、暗証番号が推測できるものや覗き見に弱いものである。肉筆は、字の形、大きさ、書く速さにより、大きな個性が見られるので、書くという行為に、個人認証ができるほどの特徴があるのではないかと考えた。そのため、本研究では、個人から描く速さや描き順が推測されにくく、覗き見されても再現されづらいと考えられる一筆書きを用いた方法を提案する。また、一筆書きを描く行為はタッチスクリーンのみを利用するため、既存の端末でも用いることができる手法である。
広い意味での一筆書きとは、平面から一度も離さずに線図形を描くことであるが、本研究での用いる一筆書きはそれに加えて、同じ線を二度以上なぞらない（図形の点で交差を除く）という条件を加えたものを想定している。はじめに持ち主を登録する作業として、スマートフォンやタブレット端末の持ち主が画面上に表示されている一筆書きを描き、描き始めから一定時間経過ごとにタッチしている縦と横の位置を記録する。これを複数回行い、持ち主の特徴量データを複数回測定し、持ち主の特徴量データの平均を求め、記録した持ち主の特徴量データ群の距離と平均の距離との差のうち絶対値が最大となるものを求め、認証の基準となる値を決める。

その後、一筆書きが描かれた場合、描き始めから一定時間経過ごとにタッチしている縦と横の位置を記録し距離を求める。この距離とはじめに記録した特徴量データの平均の距離との差の絶対値と決められた認証の基準となり値とを比べ、基準値の方が大きいまたは同じ場合は、持ち主と判断し認証され、小さい場合は他人と判断し認証しない。この提案手法のアルゴリズムを図 1 に示す。

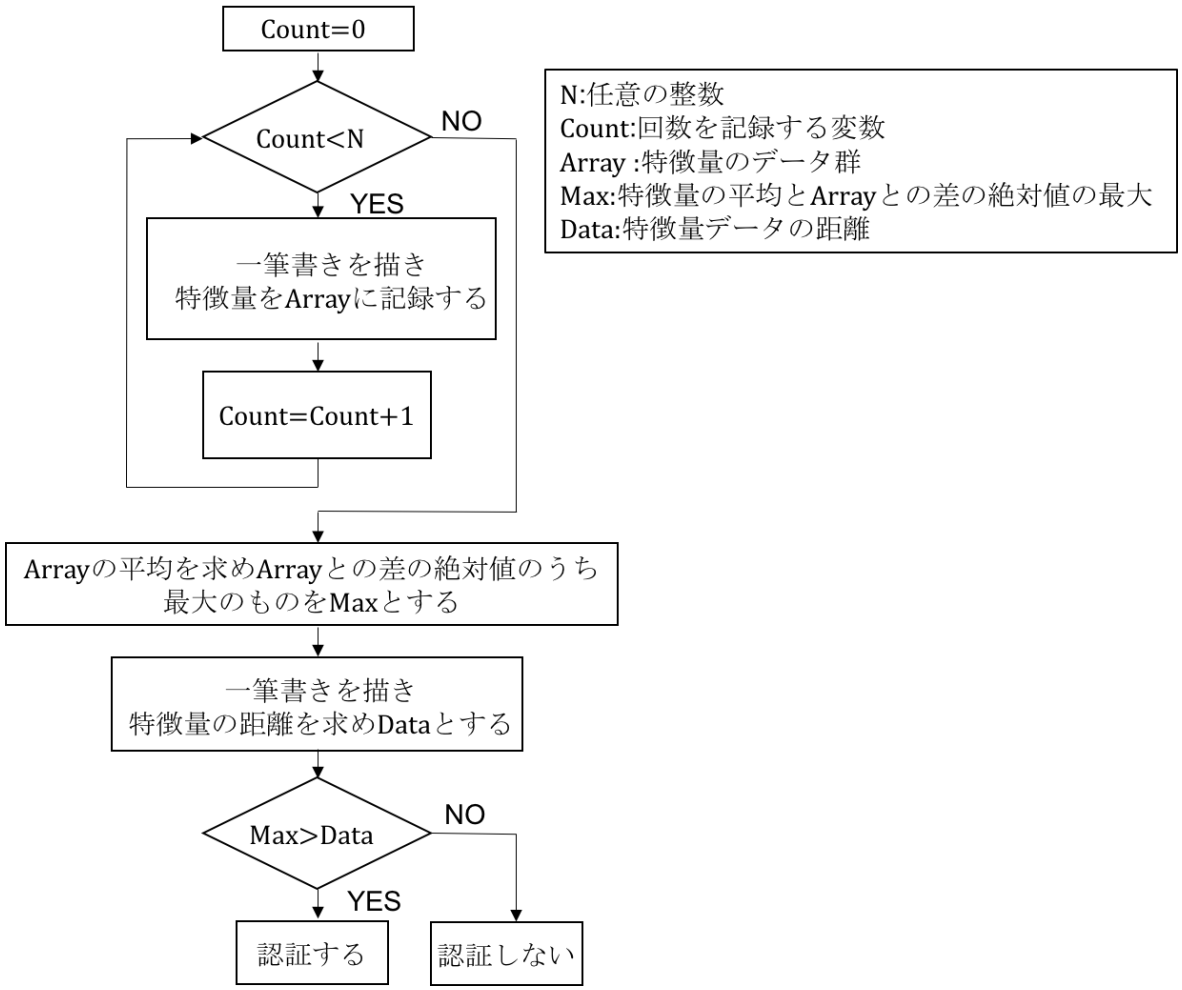


図 1:提案手法のアルゴリズム

2　実験

2.1　実験 1

2.1.1 目的

同じ人が同じ一筆書きを複数回描き、特徴量のデータを測定した場合、データのまとまりはあること、複数人が同じ一筆書きを描いた場合、他人のデータ間に違いが見られることを示し、一筆書きを描き特徴量データを測定する手法は、個人を識別する方法として利用できることを示す。

2.1.2 方法

はじめに、タブレット端末の画面に表示されている図 2 に示す一筆書き可能な図形を図上の 1、2、3、4、5、3、1、5、6、1 の順に各点を通るように描き 50ms ごとにタッチしている縦と横の位置を各 50 個、合計 100 個の値を測定し、図 3 のような形式で特徴量データとして記録する。図 3 のデータは長いため改行を行い 9 行として表しているが本来は測定 1 回につき 1 行である。図 2 の図形を描き終わるまでに 2.5s 経過した場合は、そこで測定を終了しその地点までの記録を特徴量データとする。2.5s よりも早く描き終わり指を離した場合、それ以降は指を離した場所の縦と横の位置を測定値として記録する。これを一人が 10 回連続で行い、1 セットとし、被験者 A〜F の 6 人が 3 セットずつ行う。

記録したデータを個人ごとに分け、階層的クラスタリングにより同一人物間のデータ全てが一つのクラスになる距離を確認する。次に、記録した全てのデータを自己組織化マップにマッピングを行い、他人物間のデータの関係を確認する。

特徴量データを測定するプログラムは Java で作成し、Android Studio のデバッグ機能を用いてタブレット端末でタッチされている位置をコンピュータ上に出力し記録する方法を用いる。

階層的クラスタリングは Python のライブラリである scipy を用いて特徴量データの距離を求め、デンドログラムで図示しグラフを作成す

る。自己組織化マップは som_pak-3.1 を用いて図を作成する。詳しい実験環境は表 1 に示す。

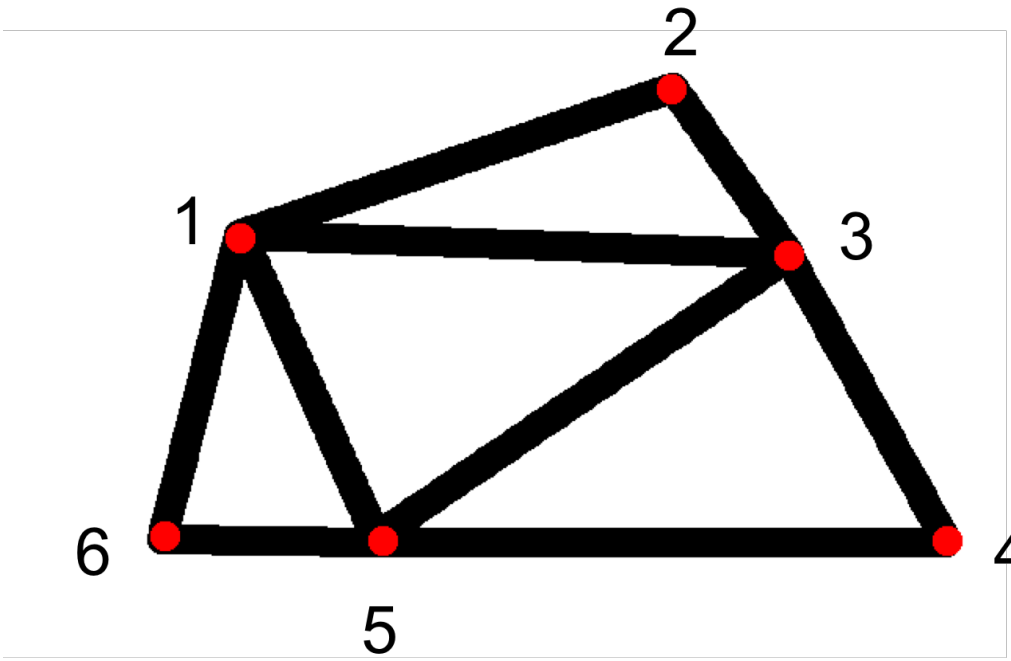


図 2:実験に用いる一筆書きの図

```
244.0 291.0 244.0 291.0 244.0 291.0 241.0 277.0 238.0 257.0 236.0
229.0 232.0 175.0 229.0 145.0 226.0 132.0 224.0 123.0 226.0 120.0
229.0 119.0 246.0 116.0 316.0 110.0 375.0 111.0 429.0 118.0 540.0
137.0 629.0 145.0 686.0 152.0 760.0 165.0 799.0 173.0 816.0 176.0
821.0 177.0 823.0 177.0 826.0 177.0 839.0 186.0 877.0 218.0 924.0
262.0 975.0 310.0 1021.0 359.0 1071.0 410.0 1113.0 456.0 1133.0
476.0 1146.0 498.0 1152.0 508.0 1154.0 513.0 1154.0 514.0 1147.0
517.0 1117.0 529.0 1076.0 547.0 1025.0 561.0 976.0 577.0 916.0 592.0
 895.0 595.0 853.0 603.0 846.0 603.0 842.0 603.0 833.0 578.0 831.0 535.0 840.0 455.0
```

図 3:実験で記録する特徴量データの形式

表 1:実験 1 の実験環境

ソフトウェア:Android Studio 2.3
タブレット端末:ICONIA TAB A200
サイズ 175×12.4×260mm
画面サイズ:10.1 インチ
OS:4.0.3
CPU:Tegra 2

2.2 実験 2

2.2.1 目的

提案する手法のアルゴリズムに従い、持ち主と他人の認証をそれぞれ行い、認証率と排除率を求め、提案手法がスマートフォンやタブレット端末のタッチスクリーンを用いた画面ロック解除の方法として用いることが可能であるということを示す。

2.2.2 方法

実験 1 で測定した 6 人分の特徴量のデータを用いて、図 4 に示す識別式と図 5 に示す自身を認証する回数を求めるアルゴリズムと図 6 に示す他人を認証しない回数を求めるアルゴリズムを元に作成したプログラムを 1000 回実行し、自身の認証と他人の認証を各 10000 回ずつ試みる。これを 6 人分行い、それぞれの 10000 回中の認証された回数と認証されなかった回数から自身を認証する確率と他人を認証しない確率を求める。実験 2 に用いるプログラムは Python で作成する。

持ち主を認証する条件式

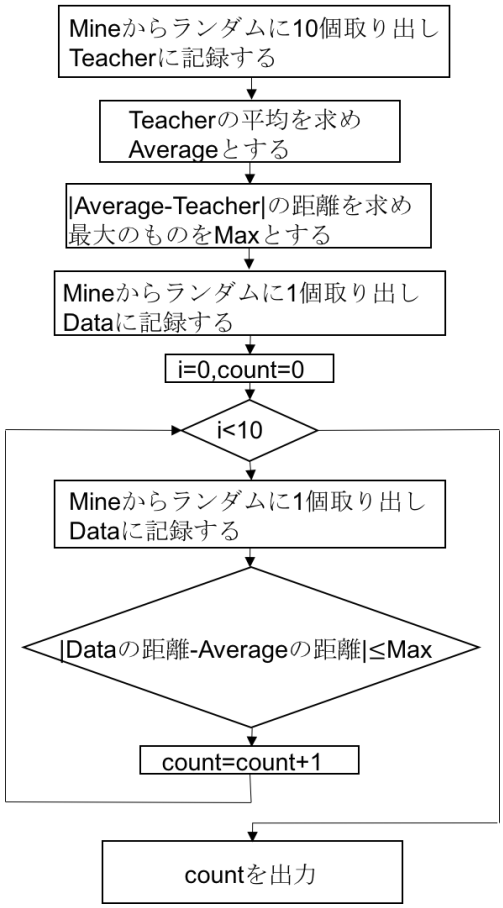
$$MAX \geq |DATA - AVERAGE| \quad (1)$$

他人を認証しない条件式

$$MAX < |DATA - AVERAGE| \quad (2)$$

ARRAY:持ち主の特徴量データの距離の集まり
AVERAGE:ARRAYの平均
MAX:ARRAYとAVERAGEとの差のうち絶対値が最大のもの
DATA:比較対象の特徴量データの距離

図 4:提案手法の識別式



Mine:持ち主のデータの集まり
Teacher:教師データの集まり
Average:教師データの平均
Max:教師データと教師データの平均との最大差の絶対値
Data:持ち主のデータから選ばれたデータ

図 5: 自身を認証する回数を求めるアルゴリズム

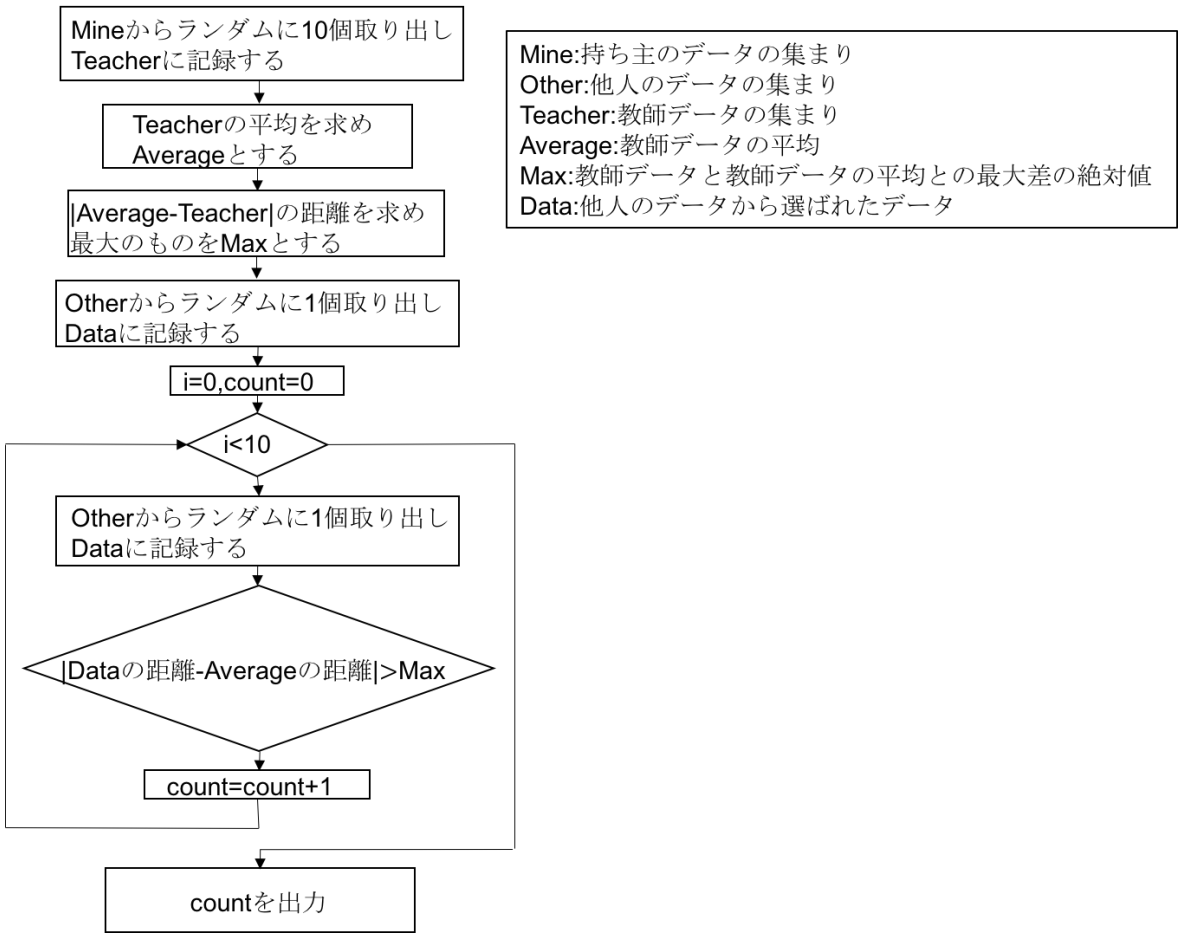


図 6: 他人を認証しない回数を求めるアルゴリズム

3　結果・考察

3.1　実験 1 の結果

3.1.1 階層的クラスタリングにより分類した結果

被験者 A～F の 6 人から測定した各 30 個ずつの特徴量データを階層的クラスタリングによりそれぞれ個人の特徴量データごとに分類した結果を図 7～図 12 に示す。各図の縦軸は距離、横軸は特徴量データの番号を示している。図 12 を見ると、個人のデータが一つのクラスとなる最大の距離は約 450、図 8 を見ると、最小は約 200 であることがわかる。

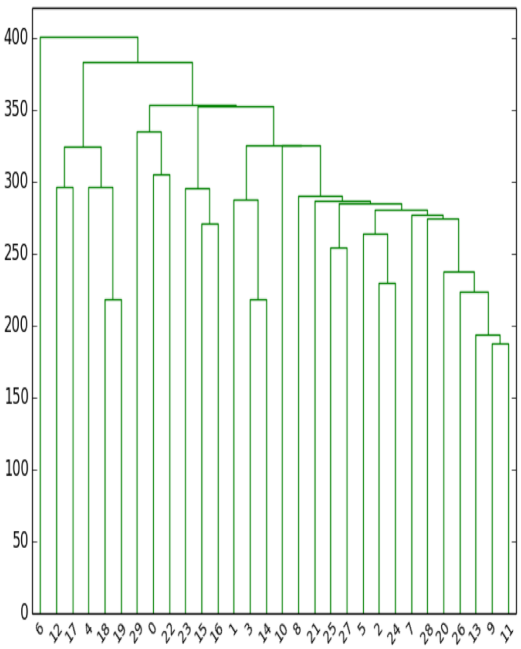


図 7:被験者 A の結果

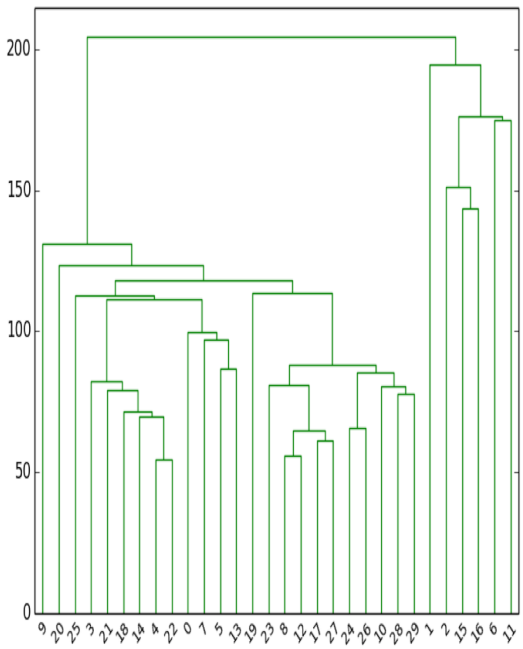


図 8:被験者 B の結果

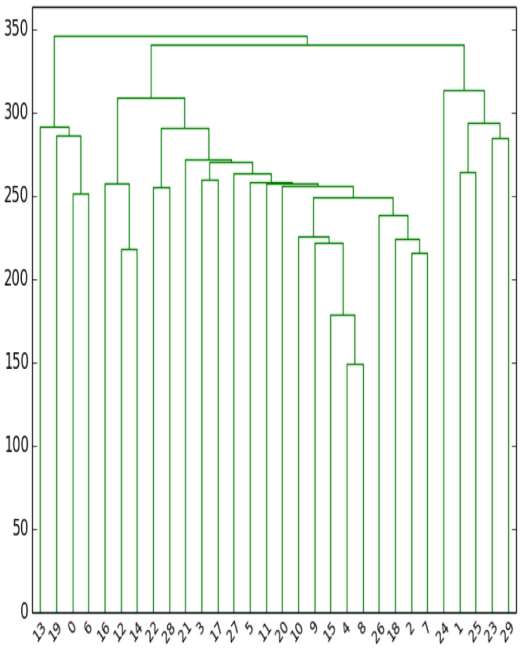


図 9:被験者 C の結果

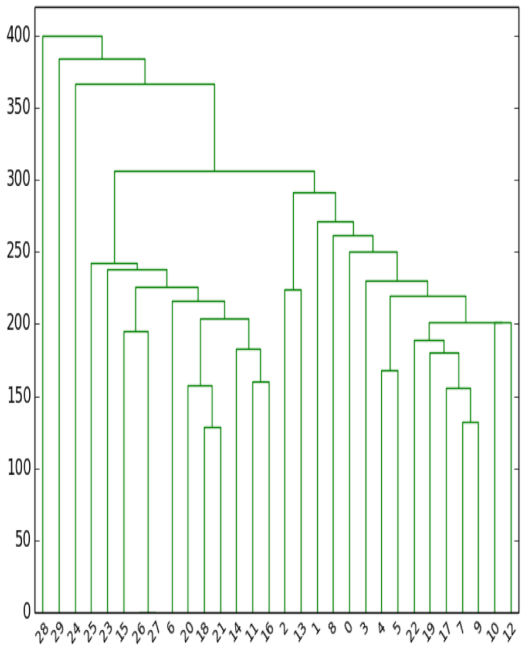


図 10:被験者 D の結果

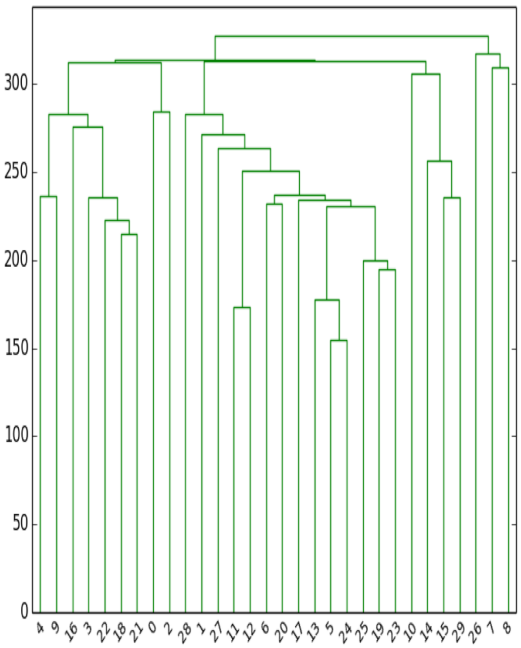


図 11:被験者 E の結果

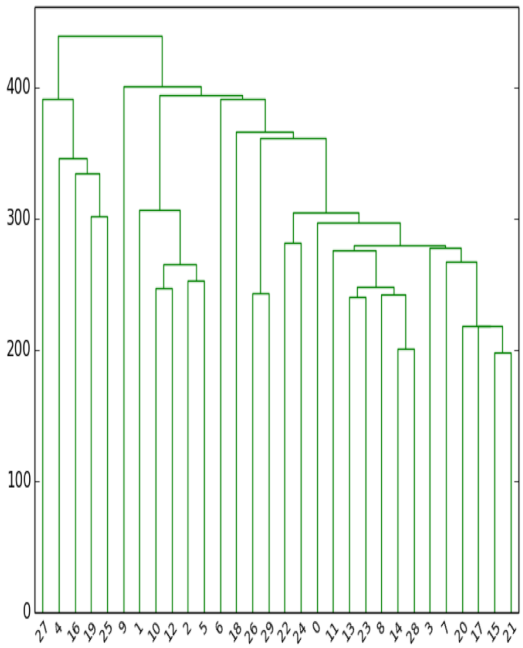


図 12:被験者 F の結果

3.1.2 自己組織化マップにより分類した結果

次に被験者 A〜F の 6 人から得た合計 180 個の特徴量データをノード数 15×15 の自己組織化マップで学習し、マッピングした結果を図 13 に示す。図中のアルファベットは被験者 A〜F を表し、次の数字は特徴量データの番号を表している。図中の六角形はノードを表しており色が濃いところは距離が離れていることを示している。図 13 を見ると被験者 B、D のデータは他の被験者のデータとの距離が大きいことがわかる。また、他の被験者のデータもまとまりがあることがわかる。

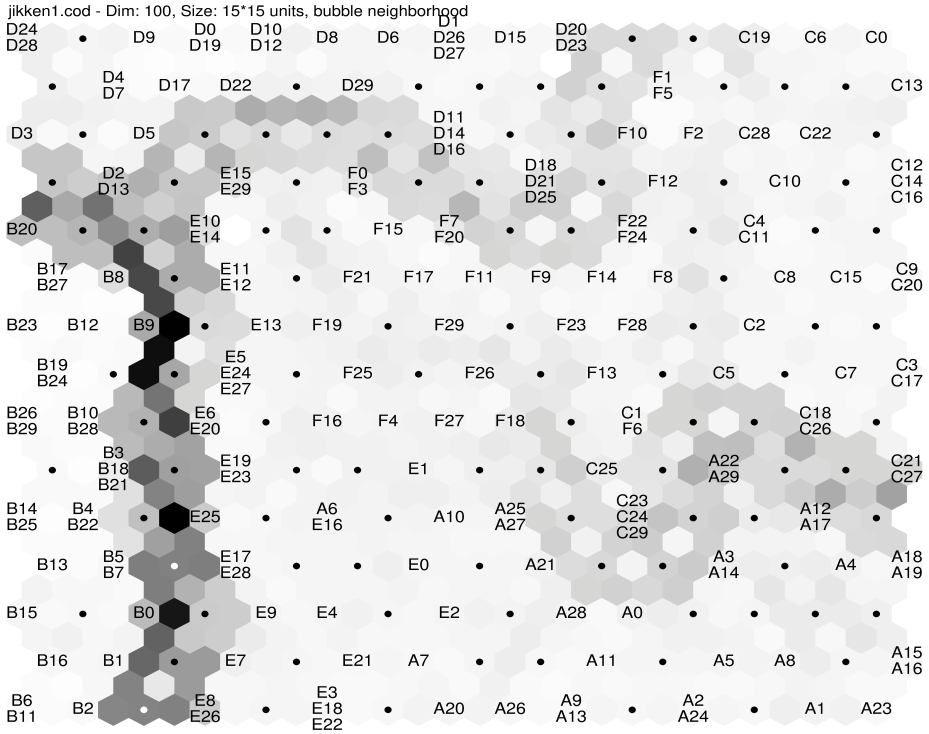


図 13:自己組織化マップでの学習結果

3.2　実験 1 の考察

被験者 1 人ずつの特徴量データを階層的クラスタリングにより分類した結果から、最も特徴量データにばらつきがある人は被験者 F で図 12 を見ると、全てのデータが一つのクラスになる距離は約 450 であり、最も特徴量データにまとまりがある人は被験者 B で図 8 を見ると、全てのデータが一つのクラスになる距離は約 200 であることがわかる。このことから、同じ人が複数回、同じ一筆書きを描いた場合、約 200〜約 450 の距離のまとまりが見られることがわかる。

また、全ての被験者の特徴量データを自己組織化マップにマッピングした結果から、被験者 A のデータは図 13 の右下、被験者 B のデータは図 13 の左下、被験者 C のデータは図 13 の右上、被験者 D のデータは図 13 の左上、被験者 E のデータは図 13 の下、被験者 F のデータは図 13 の真ん中に分類されており、違う被験者間のデータが交差している部分も少ないことから、複数人が同じ一筆書きを描いた場合、人により違う特徴が見られることがわかる。

実験 1 の結果から、同じ一筆書きを同じ人が複数回描いた場合、一定の特徴が見られ、違う人が同じ一筆書きを同じ書き順で描いた場合、その人特有の違う特徴が見られることがわかる。このことから、一筆書きを描き一定時間ごとにタッチしている縦と横の位置を記録しこれを個人の特徴量データとして用いる手法は個人を識別する方法として用いることが可能であると言える。

3.3　実験 2 の結果

図 4 の自身を認証する回数を求めるアルゴリズムと図 5 の他人を認証しないアルゴリズムを元に作成したプログラムを被験者 6 人の特徴量データに対してそれぞれ 1000 回ずつ実行し、各アルゴリズムに対して 10000 回の認証を試みた結果から自身の特徴量データを認証する確率と他人の特徴量データを認証しない確率を求め、各確率の平均と共に表としてまとめたものを図 14 として示す。

被験者	自身を認証する確率	他人を認証しない確率
A	85.98	100.0
B	90.02	96.41
C	85.75	100.0
D	86.21	100.0
E	86.39	100.0
F	85.46	100.0
平均	86.64	99.40

図 14: 自身を認証する確率と他人を認証しない確率

3.4　実験 2 の考察

被験者 A〜F の 6 人それぞれの自身を認証する確率を求めた結果から、最も高い確率は、被験者 B の 90.02%で、最も低い確率は、被験者 F の 85.46%である。全ての被験者の自身を認証する確率は約 85%から約 90%の間にまとまっており、平均を見ると 86.64%であり、自身を認証する確率は個人差が少なく、認証率は高いことがわかる。

また、被験者 A〜F の 6 人それぞれの他人を認証しない確率を求めた結果から、被験者 B を除く全ての被験者の他人を認証しない確率が 100%であり、最も低い被験者 B でも 96.41%であり、ほとんど他人を認証することがないことがわかる。また、自身を認証する確率と同じく他人を認証しない確率も個人差が少ないことがわかる。

実験 2 の結果から、提案する手法は自身の認証と他人の排除をともに、高い確率で行うことができるかつ、双方ともに確率に個人差が少なく誰でも同じように利用することが可能であると言える。

4　結論

本研究で提案したタッチスクリーン上に表示されている一筆書きを描き、特徴量データを測定することで個人を識別する手法は、特徴量データとして、一定時間ごとにタッチしている縦と横の位置を測定したものを用いることで個人を識別できた。同じ人が複数回、同じ一筆書きを描いた場合、データには一定のまとまりが見られ、複数人が同じ一筆書きを描いた場合は、それぞれ違う特徴が見られることがわかった。また、被験者 6 人の自身を認証する確率の平均は 86.64%、他人を認証しない確率の平均は 99.40%であり、双方の確率ともに高く、6 人全員の結果は平均に近く、個人により差がほとんどないことがわかった。そのため、本研究で提案した手法は個人を識別可能であり、同一人物間での再現性が高いため、スマートフォンやタブレット端末の画面ロック解除方法として利用可能である。

謝辞

本論文を作成するにあたり、ご指導いただきました三好力教授に深謝いたします。また、多忙の中、本研究において多くの助言をいただきました研究室の皆様に深く感謝いたします。

参考文献

1)総務省（2017）「平成 28 年度版情報通信白書」，
＜http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h28/html/nc252110.html＞

付録

実験 1 で用いたプログラム

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import java.util.TimerTask;
import java.util.*;
public class myTouchEvent_3 extends Activity {
    float X = 0;
    float Y = 0;
    float Xs[] = new float[50];
    float Ys[] = new float[50];
    int count = 0,num = 0;
    boolean stop = true;
    TimerTask task = new TimerTask(){
      public void run(){
          if(count < 50 && !stop){
              Xs[count] = X;
              Ys[count] = Y;
              count++;
          }
      }
    };
    Timer timer = new Timer();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        timer.schedule(task,0,50);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onTouchEvent(MotionEvent event){
        X = event.getX();
        Y = event.getY();
        String action = "";
        switch(event.getAction() & MotionEvent.ACTION_MASK) {
            case MotionEvent.ACTION_DOWN:
                action = "Down";
                stop = false;
                break;
            case MotionEvent.ACTION_MOVE:
                action = "Move";
                break;
            case MotionEvent.ACTION_UP:
                action = "UP";
                String result = "¥n";
                stop = true;
                for(int i=count;i<50;i++){
                    Xs[i] = Xs[count-1];
                    Ys[i] = Ys[count-1];
                }
                for(int i=0;i<50;i++){
                    result+=Xs[i]+" "+Ys[i]+" ";
                }
                result+="¥n";
                num+=1;
                Log.v("result"+num,result);
                count = 0;
                break;
            case MotionEvent.ACTION_CANCEL:
                action = "Cancel";
                break;
        }
        return super.onTouchEvent(event);
    }
}
```

実験 2 で用いたプログラム

```
import numpy as np
import random
import math
file = np.loadtxt("sankaku10.dat",delimiter = " ")
start = 0
correct_data = np.reshape(file[start],(1,file[start].shape[0]))
file = np.delete(file,start,0)
for i in range(29):
    tmp=np.reshape(file[start],(1,file[start].shape[0]))
    correct_data=np.concatenate((correct_data,tmp),axis=0)
    file = np.delete(file,start,0)

incorrect_data = file

def teacher_correct_select(all_data):
#正解データ全て
    all = all_data
#教師の平均と教師データの最大差
    teacher_result = 0
#教師の平均
    teacher_average_result = []
#正解
    correct_result = []
#教師データ
    selected = random.randint(0,29)
    result=np.reshape(all[selected],(1,all[selected].shape[0]))
    all = np.delete(all,selected,0)
#教師を 10 選ぶ
    for i in range(9):
        selected = random.randint(0,28-i)
        tmp=np.reshape(all[selected],(1,all[selected].shape[0]))
        result=np.concatenate((result,tmp),axis=0)
        all = np.delete(all,selected,0)
#正解を 20 選ぶ
    selected = random.randint(0,19)
    correct_result=np.reshape(all[selected],(1,all[selected].shape[0]))
    all = np.delete(all,selected,0)
    for i in range(9):
        selected = random.randint(0,18-i)
        tmp=np.reshape(all[selected],(1,all[selected].shape[0]))
        correct_result=np.concatenate((correct_result,tmp),axis=0)
        all = np.delete(all,selected,0)
#教師の平均
    teacher_average_result=result[9]
    for i in range(9):
        teacher_average_result=teacher_average_result+result[i]
    teacher_average_result=teacher_average_result/10
#差の最大
    for i in range(10):
        tmp = abs(teacher_average_result-result[i])
        tmp=tmp*tmp
        tmp=np.sum(tmp)
        tmp=math.sqrt(tmp)
        if tmp>teacher_result:
            teacher_result=tmp
    return (teacher_result,teacher_average_result,correct_result)

def incorrect_select(all_data):
    all = all_data
    result = []
    for i in range(20):
        selected = random.randint(0,29-i)
        result=np.append(result,all[selected])
        all = np.delete(all,selected,0)
    return result

correct_count = 0.
incorrect_count = 0.

for i in range(1000):
    teacher,average,correct = teacher_correct_select(correct_data)
    incorrect = incorrect_select(incorrect_data)
    for i in range(10):
        tmp = abs(average-correct[i])
        tmp=tmp*tmp
        tmp=np.sum(tmp)
        tmp=math.sqrt(tmp)
        if tmp<=teacher:
            correct_count=correct_count+1
    for i in range(10):
        tmp = abs(average-incorrect[i])
        tmp=tmp*tmp
        tmp=np.sum(tmp)
        tmp=math.sqrt(tmp)
        if tmp>teacher:
            incorrect_count=incorrect_count+1

print(correct_count/100)
print(incorrect_count/100)
```