卒業論文 2016 年度 (平成 28 年度)

AR Batting :

拡張現実を利用したバッティング練習支援システム

指導教員

慶應義塾大学環境情報学部

徳田　英幸
村井　純
楠本　博之
中村　修
高汐　一紀
Rodney D. Van Meter III
植原　啓介
三次　仁
中澤　仁
武田　圭史

慶應義塾大学 環境情報学部

佐藤 亮輔

satryo@ht.sfc.keio.ac.jp

卒業論文要旨 2016 年度 (平成 28 年度)

AR Batting : 拡張現実を利用したバッティング練習支援システム

論文要旨

近年，AR(拡張現実) と呼ばれる技術が世の中に浸透してきている．スマートフォンやウェアラブルデバイスの普及に伴い，AR を用いたアプリケーションが登場し，身近なところで利用できるようになった．それに伴い，AR を利用したスポーツ支援が登場してきている．AR を利用したスポーツ支援ができるようになると，スポーツの練習に応じた様々なアプリケーションを提供できるようになる．例えば，ユーザへ行動を指示するアプリケーションや障害物を表示してユーザに行動を判断させるアプリケーションなどが考えられる．既存の製品では，スキーで通る道を AR で表示してユーザの行動を指示している．

一方，様々なスポーツの中でも特に野球は AR で支援すべきだと考える．アマチュアの野球選手について注目してみると，特にバッティング練習は，ボールを投げる他人が必要となり 1 人では練習ができないという問題が発生している．1 人で練習する手法としては素振りとバッティングセンターが挙げられるが，素振りはボールを使わないので打撃感覚が得られず，バッティングセンターではマシンとのタイミングがつかめなかったりボールの高さ調整が不便という問題がある．また野球の練習場所として利用する場所では，球技そのものが禁止・制限されたり，他人や周囲に危害を加え，ユーザ自身も怪我する可能性があったりという問題が考えられる．

本研究では，AR を利用し場所の制限にとらわれず，1 人でバッティング練習ができる支援システム「AR Batting」を提案する．このシステムは AR でピッチャーを表示してバッティングのタイミングを養うことに特化している．本研究では，Android スマートフォンと Android Wear を使用する．Android スマートフォン側では，AR 表示モジュール，通信モジュール，音出力モジュールの 3 つのモジュールから構成され，Android Wear 側では加速度取得モジュール，通信モジュール，タイミング判定モジュールの 3 つのモジュールから構成され，Android Wear で加速度データからリアルタイムに AR とのタイミング判定を行う．AR 表示モジュールでは，ピッチャーやホームベースなどを AR で表示し，投球時にボールを表示する．通信モジュールでは，打撃時に Android スマートフォンと Android Wear との通信を行う．音出力モジュールでは，Android スマートフォンのスピーカーから打撃時に音を出力する．加速度取得モジュールでは，Android Wear の加速度センサからデータを取得する．タイミング判定モジュールでは，機械学習と閾値による 2 つの判定方法を用い，加速度データから時系列分割手法や周波数分割手法を用いて特徴量を抽出し判定を行う．

評価実験として，

キーワード

拡張現実，加速度センサ，機械学習

慶應義塾大学 環境情報学部

佐藤 亮輔

Abstract of Bachelor's Thesis Academic Year 2016

AR Batting : The Batting Support System Using Augmented Reality

Abstract

Recently, devices such as smartphones or wearable devices have various kind of sensors due to downsizing and high performance of sensors. Many people use various kind of sensors on a daily basis because of spreading of smartphones and wearable devices. Along with this, the research that records the action on a daily basis has become very popular. Mean of transportation is also a part of the studies. If we can detect transfer activities, not only we record every transfer activities ,we can develop various applications using transfer activities.

For example, the application which sets silent mode in a smartphone in train or bus and the application which plays music when we are running. In the existing research, they use various sensors such as accelerometer and microphone, GPS, and so on to detect transfer activities. The system can not detect transfer activities in underground because of using GPS, and the consumption electricity of the device is increase because of using various sensors. In addition, the system limited to one position in bags. It is not practical to detect transfer activities only using smartphone on a daily basis.

In this research, I suggest the system to detect transfer activities only using an accelerometer because of detecting transfer activities with low electricity consumption in all space and in various position of the smartphone. The system to detect transfer activities consists of accelerometer module and classification module, filtering module. The system detects transfer activities only using an accelerometer of a smartphone in real time. Accelerometer module gets acceleration data from an accelerometer. Classification module extracts feature vectors using the time domain and the frequency domain and classifies transfer activity with machine learning. Filtering module corrects transfer activity by comparing the before and after transfer activity.

I evaluated this classification accuracy of this system in a real environment that targets the 5 people, and its usefulness was verified. Classification accuracy of this system was 98.44%.

Keywords
Activity Recognition, Transport, accelerometer, Machine Learning

Keio University Faculty of Policy Management
Yuki Furukawa

目次

図目次

表目次

第 1 章

序論

本章では，はじめに本研究における背景の述べる．ついで，本研究の目的を述べる．最後に本論文の構成を示す．

1.1 背景

近年，AR(拡張現実) と呼ばれる技術が世の中に浸透してきている．スマートフォンやゲーム機器の普及により，Pokemon Go[4] やニンテンドー 3DS の AR カード [5] などの AR を用いたアプリケーションが登場し，身近なところで使用できるようになった．

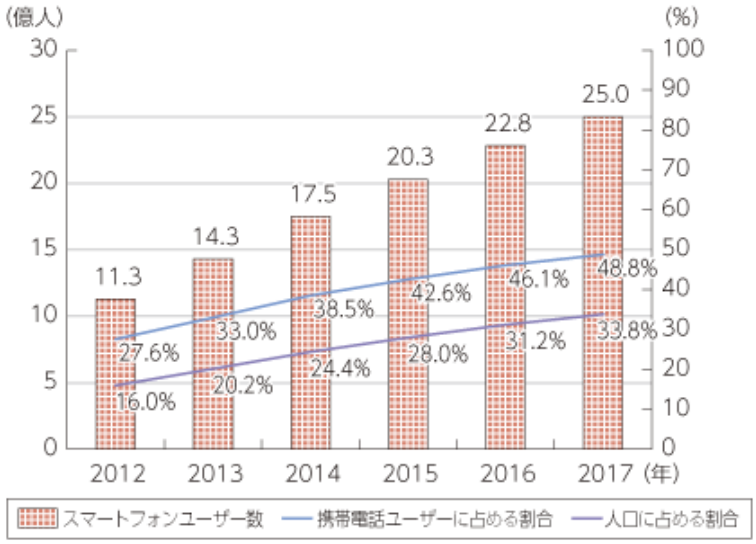

図 1.1　世界のスマートフォンユーザ数の推移（推計値）[1]

1.2 目的

本研究では，スマートフォンを用いて日常的に移動手段を判定するシステムを提案する．

1.3 構成

本論文は，本章を含め全８章からなる．本章では，本研究における背景と問題意識，目的を述べた．第２章では，本研究における移動手段の判定について整理し，問題意識を洗い出し，機能要件を導く．第３章では，本研究の機能要件を満たす移動手段判定システムを提案し，本システムの目的と特徴を述べる．第４章では，本システムでのデバイスの装着位置や機械学習の設計について述べる．第５章では，本システムの実装について述べる．第６章では，本システムで用いる特徴量と機械学習アルゴリズムの選定のため行った予備実験について述べる．第７章では，本システムの精度を評価し，考察を述べる．第８章では，本論文の結論と今後の展望について述べる．

第 2 章

関連研究

本章では，はじめに人の行動判定に関する関連研究を整理する．次に，既存研究における問題意識を洗い出す．最後に，問題意識から必要となる機能要件を導く．

2.1 人の行動判定

本節では，

2.1.1 人の動作判定

2.1.2 人の移動判定

2.2 問題意識

2.3 機能要件

2.4 まとめ

本章では，はじめに人の行動判定に関する関連研究を人の動作の判定と人の移動の判定に分け整理した．そして，既存の移動手段の判定における問題意識を洗い出し，移動手段判定システムの機能要件を導き出した．次章では，問題意識に基づく要件に対するアプローチを示し，本システムの目的と特徴を述べる．

第 3 章

AR Batting システム

本章では，第 2 章で示した問題意識から導き出した機能要件に対するアプローチを満たすシステム「AR Batting」を提案する．また，本システムの目的と特徴を述べる．

3.1 本研究のアプローチ

本研究では，スマートフォンの加速度センサのみを用いて移動手段を判定するシステムを提案する．

3.2 本システムの特徴

本システムの特徴は，GPS やマイク，角速度センサなど複数のセンサを使用せずに，加速度センサのみを使用することであらゆる場所で使用でき，省電力で移動手段を判定することが出来る．

3.3 まとめ

第 4 章

設計

4.1　本システムの設計概要

本研究では，ユーザが日常的に移動手段を判定し記録し続けるために，スマートフォン上で動作する移動手段判定システムを提案する．本システムのシステム構成図を 4.1 に示す．

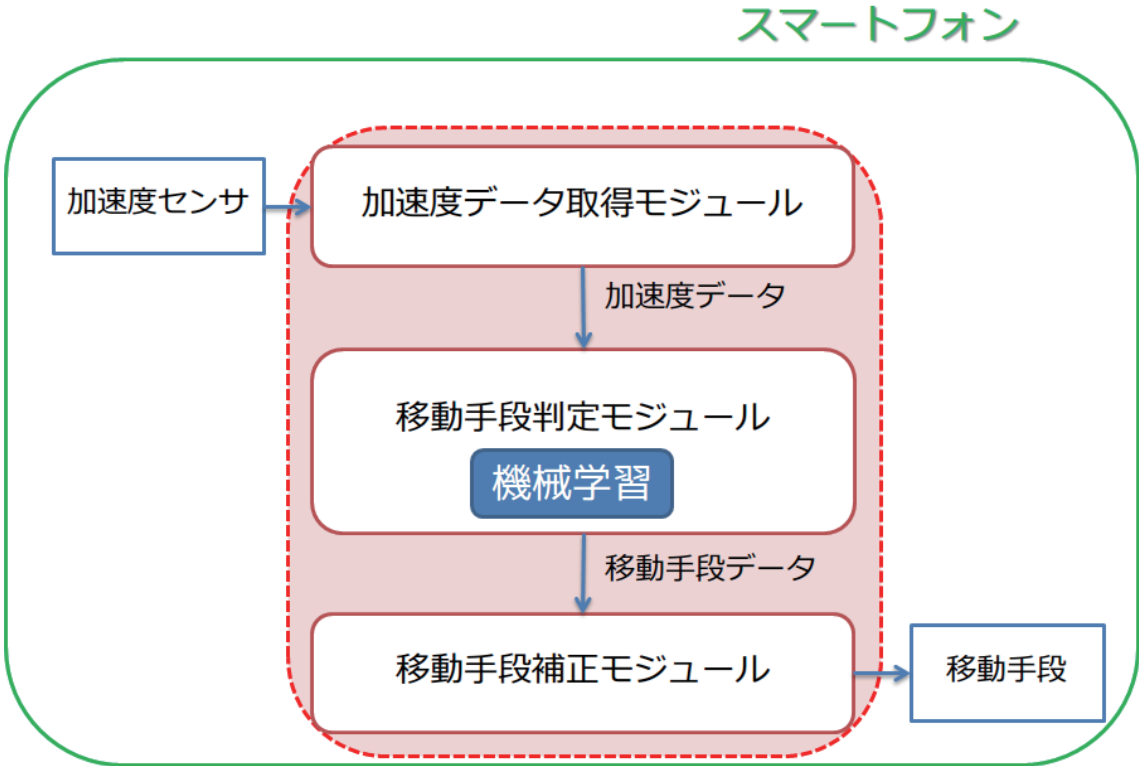

図 4.1　システム構成図

4.2 移動手段判定モジュール

4.3 移動手段補正モジュール

4.4 スマートフォンの所持位置

4.5 まとめ

本章では，移動手段判定システムの設計について述べた．次章では，加速度センサのみを用いた本システムの実装について説明する．

第 5 章

実装

本章では，加速度センサのみを用いた移動手段判定システムの実装について述べる．はじめに，加速度データを取得するために使用したセンサデバイスについて説明する．ついで，加速度センサ取得モジュール，移動手段判定モジュール，移動手段補正モジュールの実装について説明する．

5.1 使用センサデバイス

5.2 加速度センサ取得モジュールの実装

Android の Service を使用し，バックグラウンド上で加速度データを取得し続ける．加速度データが一定数溜まるごとに，移動手段を判定する．加速度センサ取得モジュールのソースコードをソースコード 5.1 に示す．

ソースコード 5.1 TransferActivityService.java

```
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Binder;
import android.os.IBinder;

import java.util.ArrayList;

import boro.life_cloud.ht.sfc.keio.ac.jp.FeatureExtractor;
import boro.life_cloud.ht.sfc.keio.ac.jp.TransferActivityClassifier;

public class TransferActivityService extends Service implements
    SensorEventListener {

    private final IBinder binder = new TransferActivityBinder();

    private SensorManager sensorManager;
```

```
    private Sensor accelerometer;

    private ArrayList<Double> xList = new ArrayList<Double>();
    private ArrayList<Double> yList = new ArrayList<Double>();
    private ArrayList<Double> zList = new ArrayList<Double>();

    private TransferActivityClassifier classifier = new
        TransferActivityClassifier();


    public class TransferActivityBinder extends Binder {

        public TransferActivityService2 getService() {
            return TransferActivityService2.this;
        }

    }

    @Override
    public void onCreate() {
        super.onCreate();

        sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER
            );

    }

    @Override
    public IBinder onBind(Intent intent) {
        // サンプリングのパラメータをSENSOR_DELAY_FASTESTに設定
        sensorManager.registerListener(this, accelerometer, SensorManager.
            SENSOR_DELAY_FASTEST);
        return binder;
    }

    @Override
    public boolean onUnbind(Intent intent) {
        sensorManager.unregisterListener(this);
        return super.onUnbind(intent);
    }

    /*
     * センサデータを取得した時に実行
     */
    @Override
    public void onSensorChanged(SensorEvent event) {

        xList.add((double) event.values[0]);
```

```
        yList.add((double) event.values[1]);
        zList.add((double) event.values[2]);


        // 一定数貯まったら移動手段を判定
        if (xList.size() >= 128) {

            // 特徴量の抽出
            ArrayList<Double> featureVector = FeatureExtractor.getfeaturevector(
                xList, yList, zList, "TD");

            // 移動手段を判定する
            String transferActivity = classifier.getTransferActivity(
                featureVector);

            // ArrayListを初期化
            xList = new ArrayList<Double>();
            yList = new ArrayList<Double>();
            zList = new ArrayList<Double>();

        }

    }


    @Override
    public void onAccuracyChanged(Sensor sensor, int i) {

    }

}
```

5.3 移動手段判定モジュールの実装

本節では，移動手段判定モジュールの実装について説明する．移動手段判定モジュールは，特徴量の抽出と機械学習を用いた移動手段の判定に分けられる．

特徴量の抽出

まず，特徴量抽出では表**??**に示した時系列分割と周波数分割の２つの特徴量を用いる．時系列分割では３軸加速度データから平均，二乗平均平方根，分散，相関係数，共分散を算出する．周波数分割では３軸加速度データから高速フーリエ変換 (Fast Fourier Transform) をする．得られたフーリエ級数の実数部分と虚数部分の係数を２乗し，パワースペクトルを抽出する．特徴量計算のライブラリに Apache Commons Math[22] を用いた．特徴量抽出のソースコードをソースコード 5.2 に示す．

ソースコード 5.2　FeatureExtractor.java

```
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.math3.complex.Complex;
import org.apache.commons.math3.stat.correlation.Covariance;
import org.apache.commons.math3.stat.descriptive.moment.Mean;
import org.apache.commons.math3.stat.descriptive.moment.Variance;
import org.apache.commons.math3.transform.DftNormalization;
import org.apache.commons.math3.transform.FastFourierTransformer;
import org.apache.commons.math3.transform.TransformType;

public class FeatureExtractor {

        // 時系列分割手法で特徴量を抽出する
        public List<Double> getFeatureVectorTD(List<Double> xList, List<Double>
            yList, List<Double> zList) {
                List<Double> featureVector = new ArrayList<Double>();

                double[] x = new double[xList.size()];
                double[] y = new double[yList.size()];
                double[] z = new double[zList.size()];

                for (int i = 0; i < x.length; i++) {
                        x[i] = xList.get(i);
                }
                for (int i = 0; i < x.length; i++) {
                        y[i] = yList.get(i);
                }
                for (int i = 0; i < x.length; i++) {
                        z[i] = zList.get(i);
                }
```

```
        // 平均
        Mean mean = new Mean();
        double xmean = mean.evaluate(x);
        double ymean = mean.evaluate(y);
        double zmean = mean.evaluate(z);
        featureVector.add(xmean);
        featureVector.add(ymean);
        featureVector.add(zmean);

        // 二乗平均平方根
        featureVector.add(Math.sqrt(xmean * xmean + ymean * ymean + zmean
            * zmean));

        // 分散
        Variance variance = new Variance();
        double varx = variance.evaluate(x, xmean);
        double vary = variance.evaluate(y, ymean);
        double varz = variance.evaluate(z, zmean);
        featureVector.add(varx);
        featureVector.add(vary);
        featureVector.add(varz);

        // 共分散
        Covariance covariance = new Covariance();
        double covxy = covariance.covariance(x, y);
        double covyz = covariance.covariance(y, z);
        double covzx = covariance.covariance(z, x);
        featureVector.add(covxy);
        featureVector.add(covyz);
        featureVector.add(covzx);

        // 相関係数
        featureVector.add(covxy / Math.sqrt(varx * vary));
        featureVector.add(covyz / Math.sqrt(vary * varz));
        featureVector.add(covzx / Math.sqrt(varz * varx));

        return featureVector;
    }

    // 周波数分割手法で特徴量を抽出する
    public List<Double> getFeatureVectorFD(List<Double> xList, List<Double>
        yList, List<Double> zList) {
        List<Double> featureVector = new ArrayList<Double>();

        double[] x = new double[xList.size()];
        double[] y = new double[yList.size()];
        double[] z = new double[zList.size()];

        for (int i = 0; i < x.length; i++) {
```

```
                x[i] = xList.get(i);
            }
            for (int i = 0; i < x.length; i++) {
                y[i] = yList.get(i);
            }
            for (int i = 0; i < x.length; i++) {
                z[i] = zList.get(i);
            }

            FastFourierTransformer fft = new FastFourierTransformer(
                DftNormalization.STANDARD);

            Complex[] xcomp = new Complex[x.length];
            Complex[] ycomp = new Complex[y.length];
            Complex[] zcomp = new Complex[z.length];

            // フーリエ変換を行う
            xcomp = fft.transform(x, TransformType.FORWARD);
            ycomp = fft.transform(y, TransformType.FORWARD);
            zcomp = fft.transform(z, TransformType.FORWARD);

            // パワースペクトルを抽出
            for (int i = 1; i <= xcomp.length/2; i++) {
                featureVector.add(xcomp[i].abs());
            }
            for (int i = 1; i <= ycomp.length/2; i++) {
                featureVector.add(ycomp[i].abs());
            }
            for (int i = 1; i <= zcomp.length/2; i++) {
                featureVector.add(zcomp[i].abs());
            }

            return featureVector;
        }

}
```

5.4 まとめ

本章では，加速度センサのみを用いた移動手段判定システムの実装について述べた．次章では，特徴量の選定と機械学習アルゴリズムの選定を目的とした予備実験について述べる．

第 6 章

予備実験

本章では，加速度センサのみを用いた移動手段判定システムで使用する、特徴量の選定と機械学習アルゴリズムの選定を目的として行った予備実験について述べる．

6.1 予備実験の概要

6.1.1 目的

6.1.2 手順

加速度データの収集

6.2 予備実験結果

6.3 考察

予備実験での分類精度

6.4 まとめ

次章では，評価用データを用いた評価実験について述べる．

第 7 章

評価

本章では，加速度センサのみを用いた移動手段判定システムの評価実験の概要を説明し，実験結果を示す．最後に，実験結果から得られた結果をもとに，本研究で提案した移動手段判定システムについての考察を行う．

7.1 評価実験の概要

7.1.1 目的

7.1.2 手順

評価実験手順は以下の通りである．

評価実験手順

1. 加速度データの収集
2. 特徴量の抽出
3. 機械学習による分類

加速度データの収集
特徴量の抽出
機械学習による分類

7.2 結果

7.2.1 教師データがあるユーザの場合

7.2.2 教師データがないユーザを追加した場合

7.2.3 教師データを作成した端末と異なる端末に変更した場合

7.3 考察

教師データがあるユーザの分類精度
移動手段補正モジュールで補正できない場合
教師データがないユーザの分類精度
教師データを作成した端末と異なる端末に変更した場合の分類精度

7.4 まとめ

本章では，加速度センサのみを用いた移動手段判定システムの評価実験を行った．評価実験の結果，教師データがあるユーザの場合の分類精度は 98.44% であった．次章では，本研究における今後の展望と本論文のまとめを述べる．

第 8 章

結論

本章では，本研究における今後の展望と本論文のまとめを述べる．

8.1 今後の展望

8.1.1 多様な移動手段の判定

8.1.2 ユーザごとの教師データの作成

8.2 本論文のまとめ

謝辞

本研究を進めるにあたり，ご指導を頂きました慶應義塾大学環境情報学部教授徳田英幸博士に深く感謝いたします．また，慶應義塾大学環境情報学部准教授高汐一紀博士，慶應義塾大学環境情報学部准教授中澤仁博士には，本論文の執筆に当たって御助言を賜りました事を深く感謝致します．慶應義塾大学徳田研究室の諸先輩方には折りに振れ貴重なご助言を頂きました．特に慶應義塾大学大学院政策・メディア研究科米澤拓郎特任助教，慶應義塾大学大学院政策・メディア研究科陳寅特任助教，慶應義塾大学大学院政策・メディア研究科研究員伊藤友隆氏には本論文を執筆するにあたってご指導頂きました．ここに深く感謝の意を表します．そして，慶應義塾大学大学院博士課程大越匡氏，慶應義塾大学大学院博士課程西山勇毅氏には，本研究に対し，多くの時間を割いていただき，ご指導を頂きました．ここに深く感謝の意を表します．

また，Life-Cloud 研究グループにおいて，様々な助言をしていただいたり，実験にご協力くださったりした佐藤翔野氏，佐々木航氏，礒川直大氏，小渕幹夫氏に深く感謝致します．

そして，数少ない同学年として，研究活動に切磋琢磨した

に深く感謝致します．

最後に，大学 4 年間に渡る生活を支えてくれた家族に感謝致します，

2016 年 12 月 6 日

佐藤 亮輔

参考文献

[1] 総務省. 平成 26 年度版情報通信白書.
http://www.soumu.go.jp/johotsusintokei/whitepaper/h26.html.

[2] ProtoGeo. Moves - activity tracker for iphone and android.
http://www.moves-app.com/.

[3] Samsung. Galaxy nexus.
http://www.samsung.com/jp/consumer/mobilephone/smartphone/docomo/SGH-N044TSNDCM.

[4] Nike, Inc. Nike fuelband se. nike.com (jp).
http://www.nike.com/jp/ja_jp/c/nikeplus-fuelband.

[5] Fitbit Inc. Fitbit.
http://www.fitbit.com/jp.

[6] Google. Android ware.
http://www.android.com/wear/.

[7] foo.log inc. Android ware.
http://www.foodlog.jp/.

[8] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, Vol. 12, No. 2, pp. 74–82, March 2011.

[9] 倉沢央, 川原圭博, 森川博之, 青山友紀. センサ装着場所を考慮した 3 軸加速度センサを用いた姿勢推定手法. 情報処理学会研究報告. UBI, [ユビキタスコンピューティングシステム], Vol. 2006, No. 54, pp. 15–22, may 2006.

[10] 品川佳満, 谷川智宏, 太田茂. ＜原著＞加速度センサを用いた人間の歩行・転倒の検出. 川崎医療福祉学会誌, Vol. 9, No. 2, pp. 243–250, dec 1999.

[11] 北沢俊二, 工藤誠一, 小板橋竜雄, 芳川美代子, 島田享久, 白鳥典彦, 富川義朗, 森泉哲次, 須山聡. 人間の行動パターンを識別するための腕時計型インテリジェントモジュールの開発. 長野県工業試験場研究報告, No. 22, pp. 47–50, 200210. Development of Wrist-Watch Type Equipment to Distinguish Motion Patterns in Human.

[12] 小林亜令, 岩本健嗣, 西山智. 釈迦:携帯電話を用いたユーザ移動状態推定方式. 情報処理学会論文誌, Vol. 50, No. 1, pp. 193–208, jan 2009.

[13] 山崎亜希子, 五味田啓. 加速度センサ等を用いた移動状態判定方式の検討. 全国大会講演論文集, Vol. 70, No. 3, pp. 39–40, mar 2008.

[14] 池谷直紀, 菊池匡晃, 長健太, 服部正典. 3 軸加速度センサを用いた移動状況推定方式. 電子情報通信学会技術研究報告. USN, ユビキタス・センサネットワーク, Vol. 108, No. 138, pp. 75–80, jul 2008.

[15] Zhixian Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer. Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach. In *Wearable Computers (ISWC), 2012 16th International Symposium on*, pp. 17–24, June 2012.

[16] Judea Pearl. Bayesian netwcrks: A model cf ‘ self-activated memory for evidential reasoning. 1985.

[17] John Ross Quinlan. *C4. 5: programs for machine learning*, Vol. 1. Morgan kaufmann, 1993.

[18] John G Cleary, Leonard E Trigg, et al. K^*: An instance-based learner using an entropic distance measure. In *ICML*, pp. 108–114, 1995.

[19] Leo Breiman. Random forests. *Machine Learning*, Vol. 45, No. 1, pp. 5–32, 2001.

[20] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING, 1998.

[21] Google. Sensormanager — android developers.
`http://developer.android.com/reference/android/hardware/SensorManager.html`.

[22] The Apache Software Foundation. Math - commons math: The apache commons mathematics library.
`http://commons.apache.org/proper/commons-math/`.

[23] The University of Waikato. Weka 3 - data mining with open source machine learning software in java.
`http://www.cs.waikato.ac.nz/ml/weka/`.

[24] HTC Corporation. Htc sensation.
`http://www.htc.com/us/smartphones/htc-sensation/`.