卒業論文

環境・生体データからの
勾配・制約を考慮した粒子群最適化による
行動パターン解析

Behavior pattern analysis by particle swarm optimization considering gradient ・ constraint from environmental・biological data

富山県立大学　電子・情報工学科
1515050　山本 聖也
指導教員　奥原 浩之 教授
平成31年2月12日

目 次

図一覧

表一覧

記号一覧

以下に本論文において用いられる用語と記号の対応表を示す.

用語	記号
クラスター	X，Y
クラスター内での重心とサンプルとの距離の 2 乗和	$\mathrm{L}(X)$，$\mathrm{L}(Y)$
クラスターの重心とクラスター内の各サンプルとの距離の 2 乗和	$\mathrm{L}(X \cup Y)$
入力データベクトル	x
出力層のニューロンの番号	i
参照ベクトル	m_i
勝者ニューロン	c
勝者ニューロンとの距離によりガウス関数で減衰する係数	h_{ci}
i 番目のニューロンの出力層上での位置	r_i
勝者ニューロンの出力層上での位置	r_c
学習回数	t
学習率係数	$\alpha(t)$
学習半径	$\sigma^2(t)$
位置	x
速度	v
運動量	w
0 から 1 の乱数	r
調整パラメータ	c
各個体の過去の最良個体	x_{db}
集団中の最良個体	x_{gb}
ステップ幅	ϕ
粒子	P
固有値	λ
PSO の調整パラメータ	$a, \alpha, \beta, \gamma, \delta$
ニューラルネットワークのダイナミクスに由来する新しい行列	X_μ
勾配情報	∇E
個体	p
個体 p の内部状態量	u^p, v^p
サンプリングパラメータ	ΔT
ステップ数	k
クラスタの数	K
クラスターの中心	C
クラスターに属するデータの中心	n
粒子のクラスター重心ベクトル	V
データセット内の最大の特徴量	R_{max}
クラスタへのパターンの割り当てを表す行列	M

第1章

はじめに

§ 1.1　本研究の背景

近年，スマートフォンを始めとする様々なデバイスを身に着けることが一般的であり，情報技術の急速な発展が著しい．中にはウェアラブルデバイスと呼ばれるものがある．それらは個人の細やかな行動をデータとして蓄積し，記録しアプリケーションを介することにより快適なサービスをユーザーに提供するとともに，健康面の管理なども行ってくれるものも存在する．このようにスマートフォンやウェアラブルデバイスを使用して取得した行動のデータは，個人の生活に活かしたり，社会に活かしたりできると考えられている．

スマートフォンやウェアラブルデバイスの全地球測位システム (Global Positioning System, Global Positioning Satellite : GPS) から個人の位置情報を取得，解析するアプリケーションが多く存在し，スケジュール情報と合わせることによりコミュニケーションツールとして活用されたり [1]，受容性の高いライフログの研究が行われている [2]．また，気温や湿度などのユーザーの周囲の環境情報と，ユーザー自身の体温や心拍数などの生体情報をログとして蓄積することにより，運動時の快適さ [3]，運転状態推定技術の開発 [4]，人体活動のモニタリングシステム [5] など，ユーザーの役に立つ情報の提示や，サービスの開発を行うことができる．

今の日本は高齢化問題，災害大国といった問題を抱えている．高齢化が進めば常時健康管理が必要な高齢者も多くなるだろう．そこでウェアラブルデバイスから環境・生体情報を管理することにより，高齢者が誤って引き起こしてしまう事故の事前の対策，看護師を始めとした医療従事者の業務の軽減が可能となる．災害時には位置情報を活用した避難システムを開発することにより，災害に対して事前に対策を講じることができる．

しかし環境ログや生体ログといったものは，データによっては精密な個人情報が含まれるため，不安や嫌悪感を感じる場合もあり，情報漏えいのリスクへの警戒など，技術面とは異なった問題も存在している．[6]．また，手動でライフログデータを取得するアプリケーションも多く，未だライフログの受容性は改善の余地がある．

環境・生体ログデータを取得する際にユーザーの負担を軽減するためにリアルタイムかつ迅速なウェアラブルデバイスの開発が必要であり，また従来法よりも有効な解析手法の

提案を行う必要があると考えられる．開発するデバイスは一般的なデバイスよりも多くのデータを取得できるべきであり，膨大なデータであっても高い精度の行動の識別が行えることが理想である．

§ 1.2　本研究の目的

本研究は，環境・生体ログシステムの開発，その情報をもとにした行動パターンの類似性やイベント性を検出する．一般的に使用されているウェアラブルデバイスやアプリケーションについて述べ，問題点を考察したうえでシステムの開発を行う．開発するシステムは無線通信から自動でのデータの蓄積を行うことにより，ユーザーへかかる負担を少なくしている．また，システムの開発にあたって，マイコンと多くのセンサを使用している．取得したデータはリアルタイムでの管理を行えるように，ブラウザ上で折れ線グラフのリアルタイムプロットを行えるシステムも加えて開発する．

行動パターンの類似性やイベント性の検出には従来手法によるクラスタリングを行う．従来手法によるクラスタリングではテキストデータが用いられているが本研究で扱うデータは数値データであるため，テキストではなく数値のクラスタリング結果を示す．示す結果は，自己組織化マップ（Self Organizing Maps : SOM），階層的クラスター分析，多次元尺度構成法（Multi Dimensional Scaling： MDS），対応分析，共起ネットワークを行い，読み取りを行うことで環境・生体ログデータの類似性やイベント性を考察擦る．

従来よりも優れた解析手法の開発を目的として，粒子群最適化 (Particle Swarm Optimization : PSO) に勾配・制約を考慮したハイブリッド的な応用手法を定式化するとともに有効性を示す．またその手法を用いた PSO のクラスタリング手法を提案する．

また PSO クラスタリングの比較対象として，自己組織化マップ（Self Organizing Maps : SOM），階層的クラスター分析，多次元尺度構成法（Multi Dimensional Scaling： MDS），対応分析，共起ネットワークを行い，読み取りを行うことでライフログデータの類似性やイベント性を考察したうえで，提案手法のアルゴリズムを示す．

§ 1.3　本論文の概要

本論文は次のように構成される．

第 1 章：本章 第 1 章では，本研究の概要と目的について説明した．

第 2 章 第 2 章では，現状のライフログ・ライフログアプリケーションの問題や特徴について説明する．また，データの取得に用いるデバイスやセンサについて説明する．

第 3 章 第 3 章では，行動識別についての研究や，行動識別のための分析手法について説明する．また，分析から考えられる類似性やイベント性について説明する．

第 4 章 第 4 章では，一般的な PSO に勾配情報を組み込んだハイブリッドな PSO についての定式化について述べる．また PSO を用いたクラスタリングについて解説し，提案手法のアルゴリズムを説明する．

第 5 章 第 5 章では，まとめと今後の課題を述べる．

第2章

ライフログと各種センサ

§ 2.1　現状のライフログ

ライフログ（lifelog) とは，人間の活動（life) の記録（log) であり，センサーなどで個人の活動に関するログを取得する行為が，ライフログの語源と考えられている [7]．本研究では，この行為をライフログとし，個人の行動履歴に基づいて生み出されるビッグデータのことをライフログデータと呼ぶこととする．また，ライフログに関して，長時間の記録や膨大なデータが必要という定義はない．

ライフログデータを取得・活用できるアプリケーションとして，iphone の専用アプリ「ヘルスケア[1]」がある (図 2.1 参照)．このアプリケーションはアクティビティ[2](図 2.2 参照) から，どれほど歩いたかという歩数や消費エネルギー等のライフログデータを取得し，ユーザ自身が健康管理に生かすことができる．また，自動で位置情報をマップにマッピングできる「マッピング - GPS ログまとめて全部記録[3]」(図 2.3 参照) や，手動でマッピングする「Swarm[4]」(図 2.4 参照) というアプリケーションがある．このアプリケーションは行動の記録を取ることができるため，日々の生活や旅行の記録として使用できる．上記のアプリケーションは GPS のアクセス許可が必要であり，上記以外のライフログアプリケーションも GPS を必要とすることが多い．

ライフログに関する既存研究として，スマートフォンから得られる位置情報履歴や写真撮影履歴，ツイートを使用したライフログデータから行動特徴抽出・イベント検出を行う研究が行われている [8] [9]．取得したライフログデータの解析を行うことで，ユーザー自身の健康管理や学習 [10] に生かすだけではなく，ビジネスとしてターゲティング広告に生かすこともできる．ライフログは，様々な視点からライフログデータの比較を行うことで，個人や社会に利用できるという価値があると考えられる．

しかし，現状のライフログには，大きくわけて二つの問題があると考えられる．一つ目

[1]https://www.apple.com/jp/ios/health/
[2]https://appllio.com/how-to-use-iphone-healthcare
[3]https://play.google.com/store/apps/details?id=org.liteapp.mat2
[4]https://play.google.com/store/apps/details?id=com.foursquare.robin

図 2.1: ヘルスケア[1]

図 2.2: アクティビティ 2[2]

はライフログの多様化問題，二つ目はライフログの煩雑問題である．

ライフログの多様化問題

ライフログを扱うサービスとしてインターネット上には多種多様なアプリケーションが登場し，様々な種類のライフログを Web 上で確認できる [12]．ユーザーの考えや気持ちを記録するブログや Twitter[5]，写真を記録する Flickr[6]，三度の食事を記録する FoodLog[7]，体や運動のデータを記録するからだログ[8]，携帯電話で写真やバーコード，睡眠時間を記録するねむログ[9]などが存在する．このように一つにライフログといっても様々な形で表すことができ，それぞれによって用いるデータは全く違うものになっている． 上記のようにライフログが多種多様になっている一方で，多くのアプリケーションを並行して使用させることはユーザーへの負担となることも考えられる．またアプリケーションが増えることによりユーザーが扱う情報量も増加し，すべての情報を正確に管理することが厳しくなってしまう．

ライフログの煩雑問題

ライフログアプリケーションの中には，意識的にライフログデータを取得しなければならないアプリケーションが存在する．このようなアプリケーションはライフログのために，ユーザーが自ら位置情報をマッピングしたり，食事風景の写真をとることを意識しなくてはならない [13]．ユーザーの主観的なライフログデータを取得できるが，ライフログデータを取得するのに手間がかかってしまうという問題を引き起こす．

[5]https://twitter.com/
[6]https://www.flickr.com/
[7]https://www.foodlog.jp/

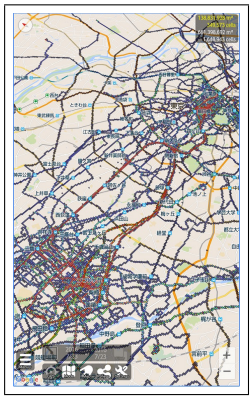

図 2.3: マッピング-GPS ログまとめて全部記録[3]

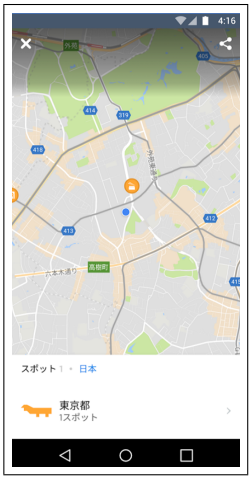

図 2.4: Swarm[4]

また，ライフログの個人情報問題に関して，株式会社ＮＴＴデータ経営研究所が 2016 年に 10 代から 60 代の男女 1059 人を対象として実施した「パーソナルデータに関する一般消費者の意識調査 [14]」という調査がある．この調査において,「企業のマーケティング等の利用目的にて，パーソナルデータを企業に提供しても良いと思うデータの条件」において，金銭や商品を受け取ることができたり，個人が特定できない状態でも，どのような条件であっても位置情報は提供したくないという人が 66.2%であり，過半数以上を占めていることがわかっている．

ライフログの多様化問題，煩雑問題という二つの問題に対し，ライフログデータを収集する上で重要であることは，一つのアプリケーションで多くのデータを取得し，ユーザーにサービスとして提示するまでの処理を手間をかけずに無意識に行うことであると考える．また示す結果に誤った情報をはじめとしたノイズが含まれてはいけない．よって画期的な手法によるライフログを正確にクラスタリングすることが重要であると考えている．

§ 2.2　各種センサとマイコンの概要

近年，スマートフォンやタブレットなどのスマートデバイスが急速に普及している．その次のデバイスとして期待されているものがウェアラブルデバイスである．ウェアラブルデバイスとは，体に装着して利用するコンピュータデバイスの総称であり，代表的なものとして，一般で使われている Apple の「Apple Watch[10]」(図 2.5 参照) や，医療や介護の現場で用いられている EMC Healthcare の「CALM-M[11]」(図 2.6 参照) などが挙げられる．

このようなウェアラブルデバイスは日常的な場面から，医療や介護といった幅広い場面で用いられている．また, ウェアラブルデバイスから得られる使用者の生体情報や個人情報は行動認識技術に活かされている．よって行動認識技術の応用範囲はとても広く，軍事 (兵

[8]https://help.goo.ne.jp/goo/g108/
[9]https://nemulog.co.jp/
[10]https://www.apple.com/jp/watch/
[11]https://prtimes.jp/main/html/rd/p/000000003.000024862.html

図 2.5: Apple Watch[10]

図 2.6: CALM-M[11]

隊，整備士)，業務 (営業マン，消防，警察，飲食店，コンビニ，警備，介護)，民生 (情報開示，記憶補助，コミュニケーション，エンタテイメント，教育) などの場面での利用を考えることができる [15].

さらに，今ではほとんどの人が持ち歩いていると言えるスマートフォンにも多くのセンサが搭載[12]されており，スマートフォンのアプリケーションを介することにより，特に必要な操作や手間なく，生体情報をはじめとした情報を得てライフログとして蓄積し，それらをユーザー自身で管理・確認することができる．

本研究では，Arduino, LLC 社の「Arduino UNO」(図 2.7 参照) や，Raspberry Pi Foundation による「Raspberry Pi 3.0」(図 2.8 参照）などのマイコンと，脈拍や体温などを測る生体センサや，気温，湿度などを測る環境センサ，これらをウェアラブルデバイスとして使用する．使用する理由としては，Arduino や Raspberrypi を用いることにより扱うことのできるセンサが幅広く，センサの追加や撤去などの調整が容易に行えるためである．

生体情報として用いるセンサは，体温[13]，心拍[14]，GSR(Galvanic Skin Response)[15]である．体温や心拍は医療や介護の現場で用いられており，ユーザーの体調管理に欠かせない．GSR は客観的に人間の心理を評価をするなどできるので感情を分析できる特徴を持つ．

また加速度[16]，温湿度気圧[17]，照度[18]，GPS[19]，人感[20]といった環境情報センサを用いる．これらのセンサと生体センサを組み合わせ，周囲の情報を取り入れることにより，ユーザーがどんな状況でどんな行動をしているか，どこでどんな行動をしているかというように，より細かい情報を得ることができる．また加速度を活かすことにより，姿勢推定や，歩行・転倒の検出など行動検知が可能となる [16] [17]．また位置情報は Google 社による「Google maps[21]」を代表として歩行，運転時の目的へのナビゲートと日常に利便性をもたらしたり，災害などの緊急時の行方不明者や徘徊老人の捜索に用いられることもある．人感センサは赤外線照度センサとも言われており，周囲の人肌以上のものを感知することができるので，

[12]https://garumax.com/smartphone-sensor
[13]http://naritaku.hatenablog.com/entry/2016/04/05/230649
[14]http://myct.jp
[15]http://wiki.seeedstudio.com/Grove-GSR_Sensor/
[16]http://akizukidenshi.com/catalog/g/gK-13010/

図 2.7: Arduino UNO

図 2.8: Raspberry Pi 3.0

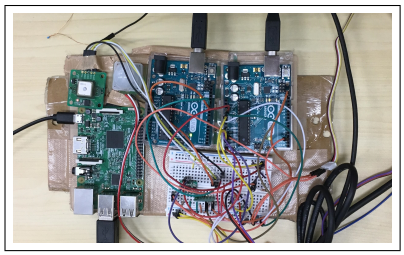

図 2.9: 制作した環境・生体ログ収集機器

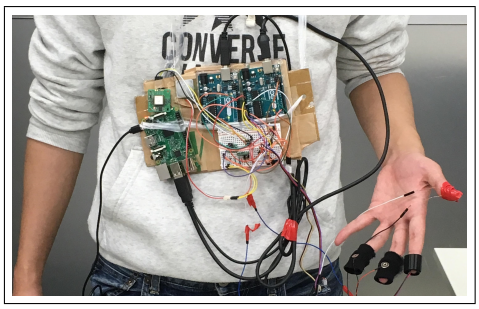

図 2.10: 実際に装着したセンサ

ユーザーが他人と接しているかどうかを判別することができる [18].

ここまで挙げたセンサをまとめると以下のようになる.

取得するデータ一覧

・環境センサ
GPS(緯度，経度，海抜)，温度，湿度，気圧，照度，人感，加速度 (3 軸)，角速度 (3 軸)，磁気コンパス (3 軸)
・生体センサ
体温，心拍，GSR

上記のセンサを Arduino を用いて計測する．Arduino に計 8 つのセンサを接続し，20 種類のデータを計測できる環境を作成する (図 2.9,2.10 参照).

[17]https://github.com/SWITCHSCIENCE/BME280/blob/master/Arduino/BME280_I2C/BME280_I2C.ino
[18]jkoba.net/prototyping/arduino/cds_practice.html
[19]https://www.petitmonte.com/robot/howto_gysfdmaxb.html
[20]http://tech.blog.surbiton.jp/arduino_motion_sensor_se-10/
[21]https://www.google.co.jp/maps/?hl=ja

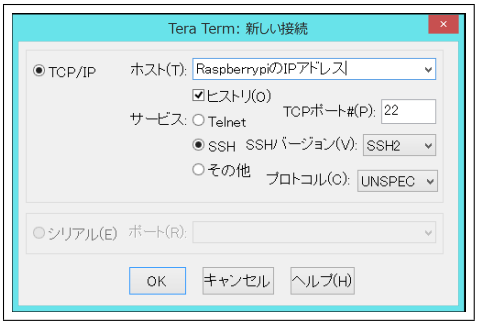

図 2.11: Tera term[22]

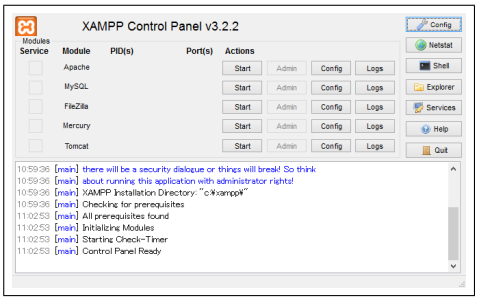

図 2.12: XAMPP[23]

§ 2.3　無線によるセンサデータ収集

ライフログを取得をするにあたって，2.2 節で挙げたマイコン (Arduino，Raspberrypi) とセンサ類を組み合わせて独自のライフログの測定環境を開発する．ライフログを取得するにはユーザーに手間がかからずスムーズかつ無線通信を利用した IoT の側面も考慮した環境を作成する必要がある．

手法としては Arduino に必要なセンサを繋ぎ，さらに Arduino と Raspberrypi を USB ケーブルで有線接続し，Arduino と Raspberrypi でシリアル通信を行う．これにより Arduino で取得したライフログが Raspberrypi に送信される．Raspberrypi には Arduino と違い Wifi 環境が整っているので無線通信を行うことができるので，Arduino から送られてきたライフログを PC に無線通信で送り，PC 内でそのライフログを蓄積する．

Raspberrypi と PC の接続は寺西高氏によって開発された Tera term[22] という Windows 上で動作する多機能端末を用いる．Windouw から Linux などにリモート接続をしようとするときによく用いられている．Tera term を起動すると IP アドレスを指定して任意の端末とリモート接続できる (図 2.11 参照)．リモート接続を終え，Raspberrypi 上で Python のプログラムを作成し，Arduino から送信されたライフログを PC に送信する．

Raspberrypi からライフログを送信するにあたって，XAMPP[23] という完全無償の MariaDB，PHP および Perl を含んだ，Apache ディストリビューションを利用する (図 2.12 参照)．これを用いてサーバーとして PC 上で Apache でローカルサーバーを作成する．これで PHP の開発環境を容易に使用することができる．

しかし，この手法では Arduino と Raspberrypi の二つのマイコンを使用しなければならないので，日常の情報を集めることが目的のライフログではユーザーに対して負担がかかってしまう．なのでライフログを測定する端末はよりコンパクトでスペースを取らないものの方が良いと考えられる．

そこで無線通信の役割を担っている Raspberrypi に代わって Arduino 上で使用することのできる Wifi モジュール ESP-WROOM-02[24] を使用する．この Wifi モジュールを用いることにより Arduino に欠けていた無線通信機能を補うことができるとともに，Raspberrypi を使用せずともよくなるので，マイコン一個分スペースを削減することができ，小型・軽量化することができる．

[22] https://ja.osdn.net/projects/ttssh2/
[23] https://www.apachefriends.org/jp/index.html

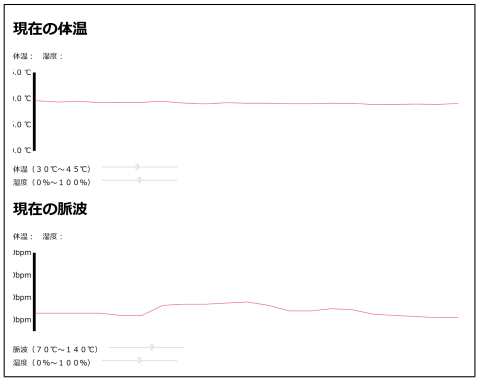

図 2.13: データの可視化[25]

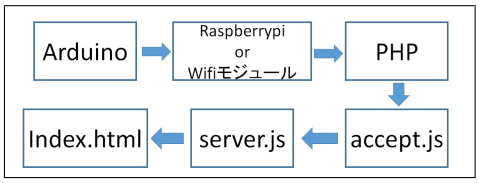

図 2.14: データフロー[26]

Wifi モジュールを用いる場合においてのライフログの処理は Raspberrypi の場合と同様に，PC 上で作成したローカルサーバーにライフログを送信する．

また 2.2 よりセンサの数を合計すると 20 次元のデータとなる．

環境・生体ログは PC 上に蓄積することができるが，データとして蓄積するだけでは意味が無い．ユーザーの情報を取得し，データをリアルタイムでサービスとして反映させることが，実際のアプリケーションでは必要である．今回はユーザーが自身の情報を視認できるように環境・生体ログの簡易的な可視化を行う．

ローカルサーバー状のデータをリアルタイムで送信，表示させるために Socket.IO[25] による通信手法を用いる．Socket.IO[25] とは web 上においてリアルタイムの通信を実現する技術の一つである．これを用いて送信されてきたデータを html で作成したページに送信する．ウェブサイトに送るために node.js[26] をインストールした PC で以下のプログラムを実行しサーバーを立てる．

このサーバーを経由してページに送信し，その結果は適当なブラウザ上で確認できる．以下にその結果の例を示す (図 2.13 参照)．この例では送信しているデータは体温と心拍である．

データの取得からブラウザ上での可視化までのデータフローを図にまとめる（図 2.14 参照）．Arduino から XAMPP[23] の Apache で立てたローカルサーバーを用いて PHP ファイルへ，そして accept.js に送信している．accept.js でシステムを 2 秒程度停止させることによりプログラムが正しく動作する．2 秒の停止がない場合では正しく動作しない．その後 PC 上で立てたサーバーから index.html というファイルに送信し，ブラウザ上でデータが確認できる．データの蓄積は PHP ファイル上で csv 形式で保存を行っている．

html でデータをグラフ化しつつリアルタイムプロットを行うには Epoch.js[27] を用いた．選んだメリットとしては，リアルタイム表示に特化していて設定が簡素であるからである．以上の手法により Arduino とセンサで取得したデータをライフログとして蓄積，表示できるようになった．

[24] https://www.amazon.co.jp/exec/obidos/ASIN/B01C8ANPYW/vlsiprograma-22/
[25] https://ics.media/entry/4320
[26] https://nodejs.org/ja/
[27] https://qiita.com/okoppe8/items/d8d8bc4e68b1da4a0a36

第3章

センサデータからの行動識別

§ 3.1　行動識別

携帯電話やウェアラブルデバイスを用いて，ユーザーが今何を行っているかという行動をライフログデータとして取得し，取得したライフログデータの解析から行動を認識することを行動認識や行動識別という．本研究では，行動識別と呼ぶことにする．

既存研究には，携帯電話の加速度センサや GPS を用いてライフログデータを取得し，走行や歩行しているなどの行動識別を行う研究 [19] や，ウェアラブルデバイスの加速度センサや GPS を利用する事で人の行うさまざまな行動を取得し，行動識別を行う研究 [20] がある．しかし，本を読んでいることであったり，料理をしていることなどの細かい動作をライフログデータとして取得することは難しい．細かい動作をライフログに組み合わせるため，手動で動作の開始・終了を記録するアプリケーション「行動の記録 (LifeLog)[28](図 3.1 参照)」や，机上に設置した Kinect™(図 3.2 参照) を用いて机上の細かい動作を認識する研究がある．しかし，この研究は机上に限っているため屋外や机上以外の行動は認識できない [21]．なお，Kinect™ は 2017 年 10 月 25 日に生産終了が公表されている．

本研究では 2 章で述べたように Arduino とセンサ類を使用して，細かい行動をライフログデータとして取得する．ユーザーの生体情報，周囲の環境情報，位置情報を取得することにより，ユーザーの体調管理やどのような行動をしているか，周囲が厚いのか寒いのか，どこにいるかなどを把握することができる．生体データと加速度データを用いての行動識別は [22] で行われている．本研究では 20 次元によるデータを扱うので従来法より，情報量が多い環境での行動識別を行うことができると考えられる．

§ 3.2　行動識別のための分析手法

本研究ではいくつかの解析手法を用いて，一定時間内の取得データを視覚的に表し，行動識別を行う．そのために，多変量解析である SOM，階層的クラスター分析，MDS，対応

[28]https://play.google.com/store/apps/details?id=com.yoko.tama.workLog&hl=ja

図 3.1: 行動の記録 (LifeLog)[28]

図 3.2: Kinect™

分析を用いてセンサデータの可視化を行う．

多変量解析を行うツールとして KH Coder[29]を使用する．KH Coder とは，テキスト型データの計量的な内容分析，もしくはテキストマイニングのためのフリーソフトウェアである [23]．無償でウェブサイトから入手でき，すべての機能をマウス操作で利用できる．また，どんな言葉が多く出現していたのかを頻度表から見ることができたり，SOM，共起ネットワークなどの多変量解析を行ったりできる [24]．KH Coder を用いて行われた研究としては，アンケートの自由回答項目・新聞記事・インタビューデータなどさまざまなデータを分析した事例がある [25]．本研究では Version 3.Alpha.11 を使用する．

データは 2 章で述べたセンサから取得する数値データを扱う．KH Coder はテキスト型データに対しての分析，もしくはテキストマイニングを目的としているため 2 節で述べたようなセンサの数値データを入力としても数値データが読み取られない．なので Arduino から取得したデータを分析を掛ける前に記録させておく必要がある．KH Coder の語の取捨選択より強制抽出する語の指定を行う．図の語は全体の一部で緯度データを指定している部分である (図 3.3 参照)．今回は机での作業時，場所を変更しての机での作業時，廊下での歩行運動を各 10 分ずつ行ったデータを用いる．

KH Coder をダウンロードする際に取得できる KH Coder3 リファレンス・マニュアルによると，KH Coder を使用した多変量解析には，まず対象のテキストファイル（もしくはエクセルファイル）を読み込む事から始める．今回はセンサデータを取得する際に作成した CSV ファイルから読み込む．読み込む CSV ファイルに事前に手動で h1 タグや h2 タグという見出しタグを設定することで，見出しごとの解析も可能である．今回はデータを取得する際にデータの先頭にそれぞれ文字をつけてそれぞれの数値との見分けがしやすくなるようにした (緯度なら LON，経度なら LAT)．そのデータの一覧を示す (図 3.4,3.5 参照)．

まず，ライフログデータの内容解析のため，階層的クラスター分析，MDS，対応分析を行う．

[29]http://khc.sourceforge.net/

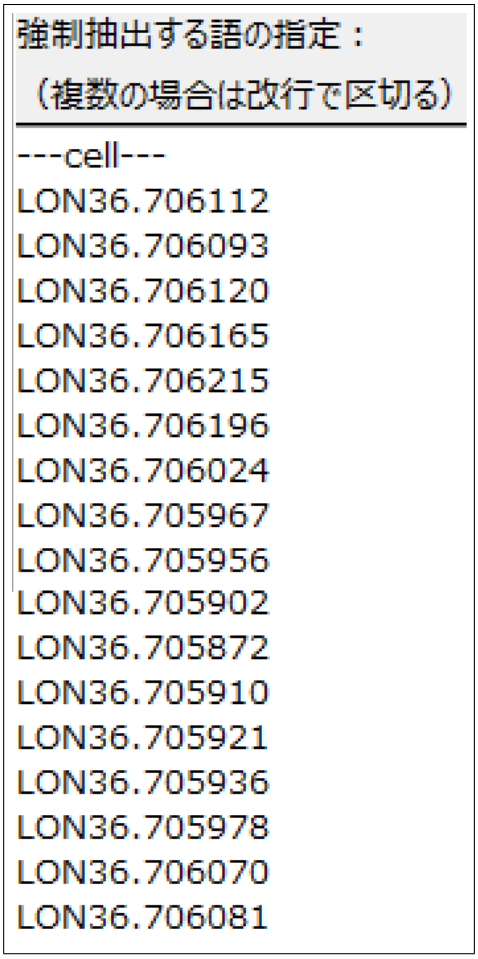

図 3.3: 強制抽出する語の指定の一部

階層的クラスター分析は，抽出語の最も似ている組み合わせから順番にクラスターにしていく方法であり，デンドログラムを表示する [26]．指定されたクラスター数に全体を分割し，その結果を色分けによって表示する．なお，KH Coder では，デフォルトの Auto では，抽出語数の平方根を四捨五入したものを用いている [27]．1 つのクラスターには関連性が高い抽出語が集まっているため，クラスターごとに集まっている抽出語を調べることでテキストデータ全体における文書の傾向や特徴を知ることができる．

階層的クラスター分析の作成方法は，まず抽出語として A, B, C, D があったとする．この時抽出語の中で最も距離の近い組み合わせを A と B とし，A と B をくくり，2 点の代表点を求める．次に，AB の重心，C，D の 3 点で，最も距離の近い組み合わせを見つける．このとき C と D が最も近いとすると，C と D をくくる．このように繰り返していくことで，デンドログラムを作成する [26] [28]．

KH Coder3 リファレンス・マニュアルによると，クラスター間の距離測定方法として，KH Coder ではウォード法を使用している．2 つのクラスター X,Y を結合したと仮定したとき，それにより移動したクラスターの重心とクラスター内の各サンプルとの距離の 2 乗和 L(X ∪ Y) と，もともとの 2 つのクラスター内での重心とそれぞれのサンプルとの距離の 2 乗和 L(X)，L(Y) の差が最小となるようなクラスターどうしを結合する手法である．ウォー

LON36.70653	LAT137.095370	HEI-75.5	T20.35	HUM34.45	PRE1019.04	rig487	zinkan0
LON36.70662	LAT137.095410	HEI-79.2	T20.28	HUM34.61	PRE1019.01	rig600	zinkan0
LON36.70657	LAT137.095470	HEI-83.8	T20.25	HUM34.63	PRE1019.11	rig490	zinkan0
LON36.70647	LAT137.095410	HEI-87.9	T20.14	HUM34.80	PRE1018.99	rig509	zinkan0
LON36.70638	LAT137.095490	HEI-91.9	T20.07	HUM34.92	PRE1019.04	rig646	zinkan0
LON36.70627	LAT137.095470	HEI-97.3	T20.02	HUM34.96	PRE1019.01	rig477	zinkan0
LON36.70620	LAT137.095630	HEI-102.1	T19.98	HUM35.10	PRE1019.00	rig668	zinkan0
LON36.70619	LAT137.095600	HEI-106.4	T19.91	HUM35.19	PRE1019.07	rig514	zinkan0
LON36.70615	LAT137.095700	HEI-106.6	T19.83	HUM35.37	PRE1018.93	rig543	zinkan0
LON36.70620	LAT137.095600	HEI-112.7	T19.84	HUM35.42	PRE1019.04	rig590	zinkan0
LON36.70625	LAT137.095570	HEI-119.2	T19.79	HUM35.43	PRE1019.01	rig450	zinkan0
LON36.70626	LAT137.095520	HEI-123.4	T19.73	HUM35.65	PRE1019.00	rig573	zinkan1
LON36.70630	LAT137.095410	HEI-128.3	T19.64	HUM35.75	PRE1019.07	rig543	zinkan1
LON36.70629	LAT137.095290	HEI-133.8	T19.67	HUM35.82	PRE1019.07	rig646	zinkan1
LON36.70628	LAT137.095250	HEI-139.5	T19.63	HUM35.83	PRE1019.04	rig578	zinkan1
LON36.70623	LAT137.095180	HEI-145.4	T19.56	HUM35.95	PRE1019.11	rig435	zinkan1

図 3.4: ラベル付けした数値データ 1

xAccl0.04	yAccl0.08	zAccl0.00	xGyro0.25	yGyro0.53	zGyro0.00	xMag8	yMag17	zMag0	BT33.8	BPM141	GSR517
xAccl0.04	yAccl0.08	zAccl0.00	xGyro0.25	yGyro0.52	zGyro0.00	xMag8	yMag17	zMag0	BT33.5	BPM121	GSR512
xAccl0.04	yAccl0.08	zAccl0.00	xGyro0.25	yGyro0.49	zGyro0.00	xMag8	yMag16	zMag0	BT33.2	BPM120	GSR508
xAccl0.04	yAccl0.08	zAccl0.00	xGyro0.25	yGyro0.49	zGyro0.00	xMag8	yMag16	zMag0	BT34.0	BPM96	GSR508
xAccl0.04	yAccl0.07	zAccl0.00	xGyro0.25	yGyro0.46	zGyro0.00	xMag8	yMag15	zMag0	BT30.6	BPM107	GSR506
xAccl0.04	yAccl0.07	zAccl0.00	xGyro0.25	yGyro0.46	zGyro0.00	xMag8	yMag15	zMag0	BT30.0	BPM107	GSR502
xAccl0.04	yAccl0.07	zAccl0.00	xGyro0.25	yGyro0.44	zGyro0.00	xMag8	yMag14	zMag0	BT29.7	BPM102	GSR501
xAccl0.04	yAccl0.06	zAccl0.00	xGyro0.25	yGyro0.42	zGyro0.00	xMag8	yMag13	zMag0	BT29.8	BPM106	GSR500
xAccl0.04	yAccl0.06	zAccl0.00	xGyro0.25	yGyro0.42	zGyro0.00	xMag8	yMag13	zMag0	BT30.1	BPM94	GSR498
xAccl0.04	yAccl0.06	zAccl0.00	xGyro0.25	yGyro0.41	zGyro0.00	xMag8	yMag13	zMag0	BT30.1	BPM64	GSR502
xAccl0.04	yAccl0.06	zAccl0.00	xGyro0.25	yGyro0.40	zGyro0.00	xMag8	yMag13	zMag0	BT30.0	BPM97	GSR502
xAccl0.04	yAccl0.06	zAccl0.00	xGyro0.25	yGyro0.39	zGyro0.00	xMag8	yMag12	zMag0	BT29.0	BPM120	GSR501
xAccl0.04	yAccl0.05	zAccl0.00	xGyro0.25	yGyro0.35	zGyro0.00	xMag8	yMag11	zMag0	BT29.5	BPM120	GSR503
xAccl0.04	yAccl0.05	zAccl0.00	xGyro0.25	yGyro0.36	zGyro0.00	xMag8	yMag11	zMag0	BT29.8	BPM98	GSR131
xAccl0.04	yAccl0.05	zAccl0.00	xGyro0.25	yGyro0.35	zGyro0.00	xMag8	yMag11	zMag0	BT29.5	BPM119	GSR41
xAccl0.04	yAccl0.05	zAccl0.00	xGyro0.25	yGyro0.33	zGyro0.00	xMag8	yMag10	zMag0	BT29.3	BPM92	GSR30

図 3.5: ラベル付けした数値データ 2

ド法は，計算量は多いが分類感度がいいため用いられることが多く，ウォード法は一つのクラスターに抽出語が順に吸収され類似するクラスターが形成される鏡効果が起こりにくいという強みがある [29] [30].

$$\Delta = L(X \cup Y) - L(X) - L(Y) \tag{3.1}$$

クラスター分析の結果を示す (図 3.6,3.7 参照)．この時クラスター数を Auto にしたためクラスター数は 10 となる．このとき併合標準（図 3.8 参照）はクラスター数を決める指標のひとつとなるのだが，縦軸の非類似度がほぼ同値であるためクラスター毎の差がこの併合水準のグラフから読み取ることができない．テキストデータを用いた場合の併合水準グラフではクラスタ毎の非類似度の差が読み取ることができる．

クラスター分析から，最も多く出現しているセンサデータが一番下のクラスタに分類されるとともに，それ以外のデータも同じように一番下のクラスタに分類されている．またほかのクラスタは分類されているデータが一つだけである．本来のクラスター分析では関

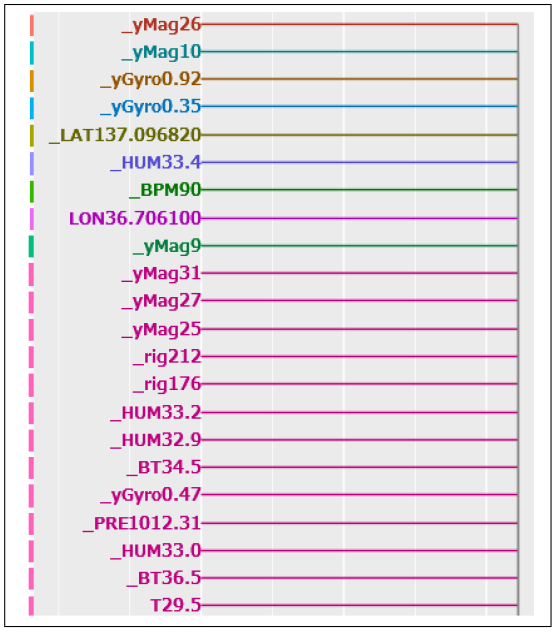

図 3.6: クラスター分析 1

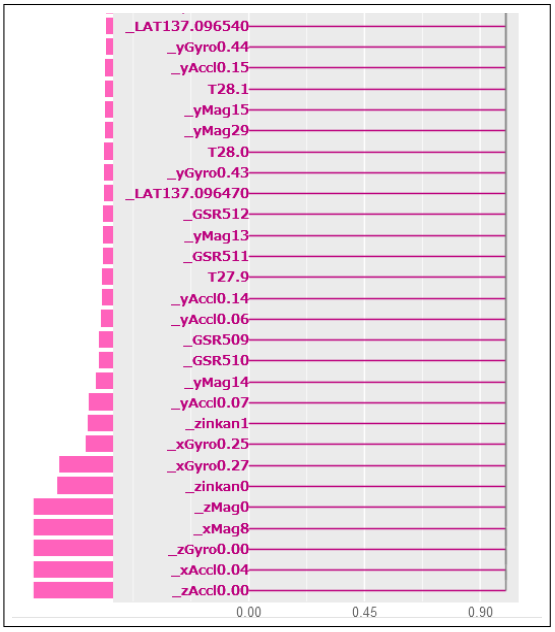

図 3.7: クラスター分析 2

連している単語や，似たような単語がそれぞれクラスターに分類され，クラスター同士の距離から語の関係を推測できる結果を示すことができる．

よってクラスター分析とその併合水準の結果からは，センサデータを正しくクラスタ毎に分類することができないことが分かる．これは KH Coder はテキストデータを想定しての多変量解析ソフトであるため，数値データを強制抽出したもののテキストデータではないためうまく分類できなかったと考えられる．またラベル付けしても数値データの非類似度がほぼ同じだったことも原因の一つと考えられる．

MDS は，抽出語間の関連性や類似性の強さをマップ上の点と点の距離に置き換えて，相対的な関係性を視覚化する手法である [31]．KH Coder では MDS の中でも最も広く利用されてきた Kruskal の非計量 MDS を使用している [32]．また，語と語の関連を見るために Jaccard 係数を使用している．Jaccard 係数とは二文章間の類似度であり，「語 A を含む」かつ「語 B を含む」文書の数を，「語 A を含む」または「語 B を含む」どちらかでも当てはまる文書の数で割った係数である [33]．MDS の結果は，相対的な位置関係だけを表わしているため軸の方向性に意味はない．

$$Jaccard\,係数 = \frac{語\,A\,と語\,B\,を含む文書}{語\,A\,もしくは語\,B\,を含む文書} \tag{3.2}$$

図 3.9 はセンサデータから出力した MDS である．クラスター分析を参考にし，クラスター数は 10 に設定した．この結果，目立つものはクラスター 01 であり，出現頻度の高い数値データが中心のクラスターである 01 に固まっている．他のクラスターを確認すると 01 のクラスターから均等に離れており，出現頻度の差はほとんどない．またどのような数値が同じクラスタにあるか確認すると，緯度，経度，心拍，加速度など，特に関係性を確認することができない．他のクラスターからそれほど離れていないクラスターがデータの中で中心的なクラスター・抽出語だと言えるのだが，他のクラスターと離れていないのはク

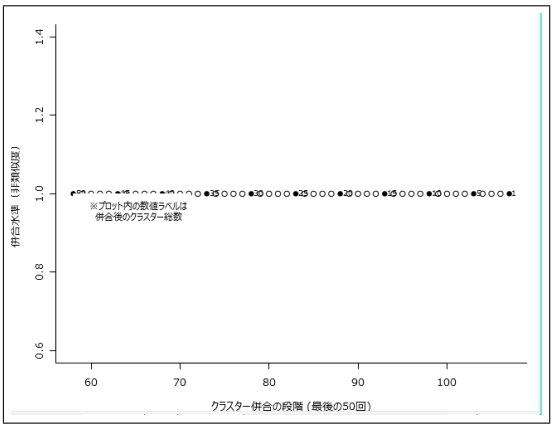

図 3.8: クラスター分析の併合標準

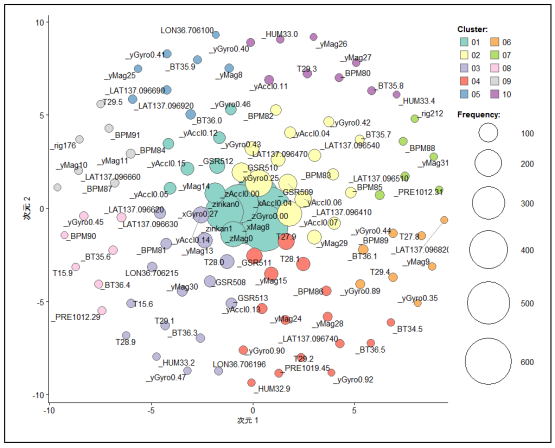

図 3.9: センサデータから作成した MDS

ラスター 01 であり，01 は頻度の高いものが集まっているだけなため，この結果からでは中心的なクラスターを確認することはできない．

対応分析は，単純な 2 次元表や多重表の行と列間の対応する測定値を分析する探索的データ解析の手法であり，分析結果として，2 次元のマップが表示される [34]．このマップで近くに位置しているものは，相対的に関連が強いということを示し，遠くに位置しているものは関連が弱いということになる．また，対応分析では，これといって特徴のない語が原点付近に密集することが多い．この時軸に表示されている数字は固有値と寄与率である．

図 3.10 はセンサデータより出力した対応分析である．本来のテキストデータを用いた場合の対応分析ではグラフ全体にある程度データが散らばり，座標 (0,0) に近い単語が類似度が高いことを示すことができる．しかしこの対応分析からでは，ほとんどのデータが座標 (0,0) の近くに位置しており，関連が強く，データとしてあまり差が出ていないことが分かる．緯度と経度のデータのうちの一つずつが座標 (0,0) から大きく離れているので，このデータは他のデータと関連が弱いというように分類された．

ライフログデータの内容解析のため，SOM を用いて，テキストデータの時系列を可視化することでテキストデータの類似性を検出する．SOM とは，ヘルシンキ大学のコホーネン教授により 1981 年頃に発表された，教師なし学習を行なうニューラルネットワークの代表例と言える解析手法である [35]．ニューラルネットワークとは，脳機能に見られるいくつかの特性を計算機上のシミュレーションによって表現することを目指した数学モデルである．つまり，人間が無意識にやっていることを機械にやらせるということである．

SOM は入力層と出力層の 2 つに分かれて競合学習を行う [36] [37]．入力層のニューロンが複数個あるが，各々のニューロンがそれぞれの次元に対応した出力を行っていると考える．入力データベクトルと呼ばれる入力層から出力層への入力を x と定義し，出力層のニューロンと入力層のそれぞれのニューロンとの結合強度は総称して参照ベクトルと呼ばれ，i を出力層のニューロンの番号とすると，m_i で表される．まず初めに，m_i の初期化を行い，入力データベクトル x を選び，入力データベクトルと各ニューロンの参照ベクトルとのユークリッド距離で出力層のニューロンを競合させる．勝者ニューロンを c とすると，式 3.3 で表される．$\arg\ \min\ f(a)$ は f(a) を最小にする a の集合であり，下側に変数がとる値の範囲を書くことが多い．

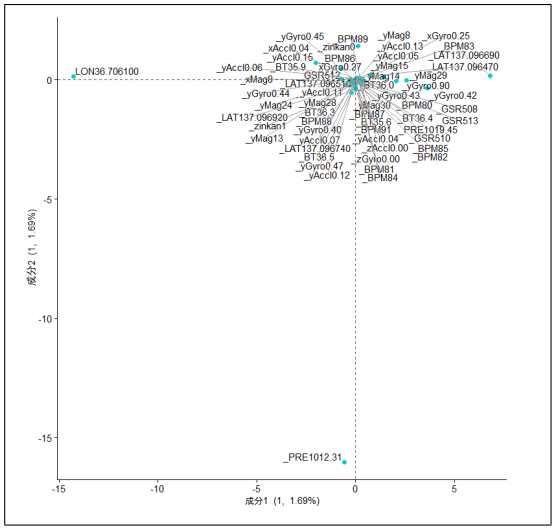

図 3.10: センサデータから KH Coder で作成した対応分析

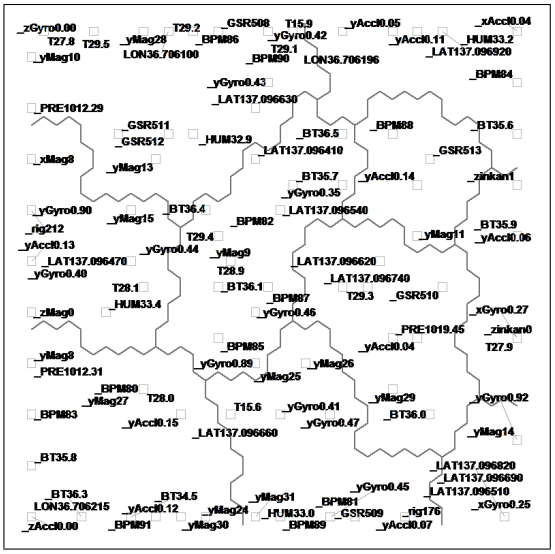

図 3.11: センサデータから KH Coder で作成した SOM

$$c = \arg\min_i \{||x - m_i||\} \tag{3.3}$$

次に，勝者ニューロンと勝者ニューロンに近いニューロンは自らの参照ベクトルと入力データベクトルを近づける学習を行うため，参照ベクトルを同様に更新させる．この時，h_{ci} は勝者ニューロンとの距離によりガウス関数で減衰する係数である．

$$m_i(t+1) = m_i(t) + h_{ci}(t) \cdot \{x(t) - m_i(t)\} \tag{3.4}$$

$$h_{ci} = \alpha(t) \cdot \exp \frac{-||r_c - r_i||^2}{2\sigma^2(t)} \tag{3.5}$$

また，SOM 作成過程ではユークリッド距離を利用している．また，KH Coder3 リファレンス・マニュアルによると，文書の長さのばらつきに左右されない形で計算を行うために，文書中における語の出現回数をそのまま使うのではなく，1,000 語あたりの出現回数に調整したものを計算に使用している．

KH Coder3 リファレンス・マニュアルによると，KH Coder の SOM の学習は，大まかな順序づけを行う段階と，微調整を行う収束段階の 2 段階で行われる．KH Coder では，1 段階目が 1,000，2 段階目が「全体のノード数を 500 倍した数値」に設定されており，全体のノード数が 40 の場合は 200,000 回である．また，各ノードがもつベクトルをウォード法で分類してクラスター化する，クラスター数は任意に決定できるため，クラスター分析などの結果からクラスター数を調整する．

作成した SOM を示す (図 3.11 参照)． 理想としては歩行時，机での作業時，気温での違いなどで分類することである．10 のクラスターに分類はできているが，別の気温が同じクラスタに分類されていたり，似たような値の生体データが別々のクラスタに分類されているなど，行動毎に識別できているクラスタリングとはいえない．

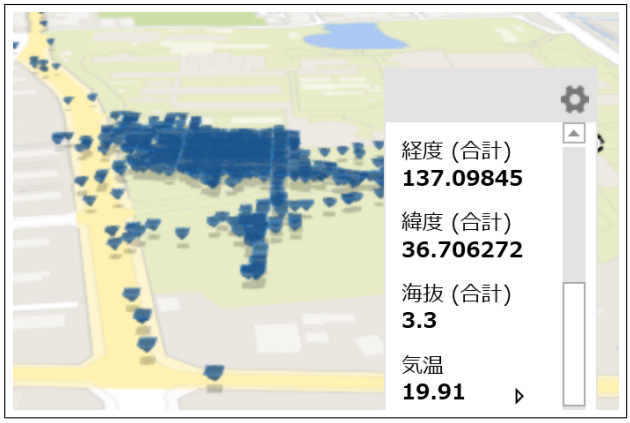

図 3.12: GPS でのマッピング

§ 3.3　類似性・イベント性

階層的クラスター分析，共起ネットワークはクラスターの分かれ方やデータを構成する単語の関係性からどのような行動があるのかがわかり，このとき予想できるクラスター数は SOM にも利用できる．MDS，対応分析はプロット点やプロット間隔から類似する行動やイベント性のある行動がわかる．SOM はクラスターの分かれ方や時系列を表すプロット順を追っていくことで行動パターンを識別し，多くのライフログの中でも類似したライフログか，特徴的でイベント性のあるライフログかわかる．

しかし行ったそれぞれの分析で行動パターンの識別がうまく行われなかった．これはセンサデータの類似度に大きな差がないため，頻度の違いしかわからない分類結果になってしまっている．

そこで取得したデータの行動パターンの分類を正しく行うため，GPS データをもとにした各センサデータのマッピングと，Excel を用いた行動パターンの分類を行う．

Microsoft Excel 2016 より追加された 3 D マップ機能を用いて収集したデータのマッピングを行う．今回扱うデータは約 5 時間，被験者の研究室内での作業時から，帰宅するまでの環境・生体データである．KH Coder の分析に用いたデータでは追加してなかったが，今回は各データへのラベル付けは行わず，年，月，日，時，分，秒の時系列データを追加する．マッピングした地図を示す (図 3.12 参照)．位置情報をマッピングしたピンにカーソルを合わせると，その場所の各データを確認することができる．これにより環境情報，生体情報を分かりやすい形で把握することができる．

次に収集したデータを Excel 上で行動パターンの分類を行う．収集したデータの環境，生体データから行動を分類し，グラフを作成する．IF 文を用いて分類を行う．まず GPS から学校と自宅の範囲を分ける．それ以外の場合は外出として分類した．次に角速度に大きな動きがあれば歩行中とした．身に着けているセンサは首からぶら下げている状態なので，動いていない場合においてはあまり大きな動きはなかったため歩行以外は滞留としてその場に留まっていると分類した．次に気温と照度の大きさからトイレに行ったときの行動を分類する．次に心拍から平静を保っているか緊張をしているかを分類した．しかし心拍センサは現状ノイズが非常に多くデータの値が高すぎる時が多く見られた．次に照度センサから明所と暗所を分類し，人感センサを反応，無反応と分類した．その分類結果を以下に

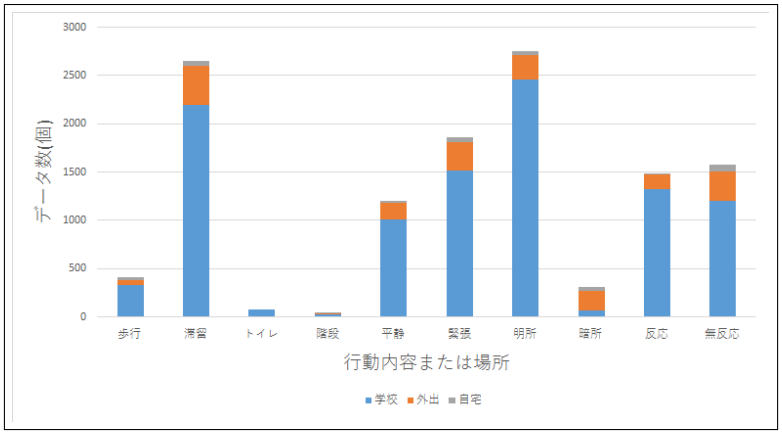

図 3.13: Excel での行動識別結果

示す (図 3.13 参照).

緯度，経度は安定したデータを収集することができたが，海抜の値が不安定だった，また体温や心拍は不適な値が多く見られたことからパラメータの調整や，用いるセンサの変更を検討する必要がある．また人感センサが被験者しかいない状態でも反応してしまうことが多々あるため，センサの装着方法などのデバイスに関する問題を把握することができた．

第4章

提案手法

§ 4.1　勾配系を考慮したPSO

Swarm Intelligence（群知能）は，鳥や魚，アリのコロニーなどのグループの行動に基づく最適化手法である． この技術の一つである粒子群最適化が開発され，様々な研究に応用されている．しかし，粒子群最適化の収束は根拠がない． 本研究では，より良い最適解を求めるための群知能とニューラルネットワークダイナミックスの新しいハイブリッド動的システムを提案した．本節では主な結果として、粒子群最適化と勾配法のメカニズムを理論的にどのように組み合わせるかを示し，今後は提案システムが客観的な環境のグローバルな情報に基づいて補間探索を実現できることを確認する．

粒子群最適化 (Particle Swarm Optimization： PSO) は，群の中の固体（粒子）が持つ最良の情報とそのグループの最適値から過去の探索から考慮した確率的最適化手法であり，ケネディ[38] が社会的行動に基づいて開発した並列進化計算技術である． 社会的方法と計算方法の両方を扱う PSO に関する標準的な研究がある [39].

近年，コンピュータサイエンスの発展は，ハードウェアとソフトウェアの有効性が顕著に表れている．その中で大規模問題の最適化の重要性はますます高めっている．ソーシャルネットワークサービスの登場により，ログやパスの問題も大規模になっている．最新のコンピュータでこれらの問題を解決するには時間がかかってしまう．

本研究では数ステップでもっとも最適な解が見つかる新しいハイブリッド動的システムを提供する．そこで連続 PSO アルゴリズムに勾配法を組み込み，定式化を行う．そこで，提案した手法の有効性を示す．その後，提案した PSO の手法を用いて取得したライフログのクラスタリング行い，その結果を示す．

PSO は群をなして移動する生物の行動を模範したアルゴリズムである．群をなす生物をモデル化し，粒子は最適化問題における候補解を示している．PSO は群の中の粒子がもつ最良の情報 (pbest) とその集団の最適値 (gbest) から過去の探索を考慮し，さらにその集団の各粒子の位置および速度を更新することによって計算される (式 4.1，4.2 参照)．以下に PSO の解説を示す (図 4.1 参照).

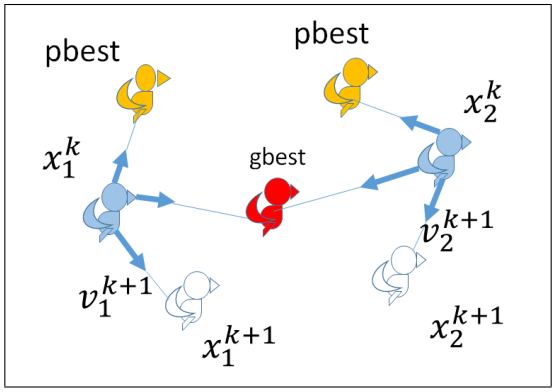

図 4.1: PSO の探索模式図

$$x_d^{k+1} = x_d^k + v_d^{k+1} \tag{4.1}$$
$$v_d^{k+1} = wv_d^k + c_1r_1(x_d^k b - x_d^k) + c_2r_2(x_g^k b - x_d^k) \tag{4.2}$$

ここで，PSO の探索模式図及び速度と位置の更新式より，pbest に向かう $c_1r_1(x_{db}^k - {x_d}^k)$，gbest に向かう $c_2r_2(x_{db}^k - {x_d}^k)$，これまでの進行方向へ向かう wv_d^k の 3 つのベクトルを合成して速度ベクトル ${v_i}^{k+1}$ を決定し，それを元に次に移動する位置 x_d^{k+1} 決定する．

PSO の探索式はランダム要素を含み，同時に最良解情報である pbest と gbest が探索に伴い変化するという時変性を有している [40]. このままの形では理論解析が困難であるので，一つの Particle に着目し，一次元の位置 x と速度 v について考え，さらに pbest と gbest を一つの点に縮約した簡略モデルが提案されている [39]. この簡略モデルは，確定的な線形時不変システムとして表現されており，その安定性を示す.

Particle$i$ に注目すると速度ベクトル v^{k+1} は以下の式のように変形できる (式 (4.3), 式 (4.4)). ステップ幅 ϕ は二つの一様乱数を足し合わせたものであり，最小値 0，最大値 $C_1 + C_2$，平均 $\frac{C_1+C_2}{2}$ の分布に従う.

$$V^k + 1 = wv^k + \phi(P - x^k) \tag{4.3}$$
$$P = \frac{\phi_1 pbest^k + \phi_2 gbest^k}{\phi_1 + \phi_2} \tag{4.4}$$

ここで，$\phi = \phi_1 + \phi_2$，$\phi_1 = C_1 rand$，$\phi_2 = C_2 rand$，さらに $y^k = p - x^k$ とおくと，式 (4.5) のように表せる．また $\phi = (C_1 + C_2)/2$ と見なすと固有値 λ は式 (4.6) のように表せる．よって λ が 1 を境にシステムの特性が安定・不安定 (収束・発散) に変化することが分かる.

$$\begin{bmatrix} v^k + 1 \\ y^k + 1 \end{bmatrix} = \begin{bmatrix} w & \phi \\ -w & 1 - \phi \end{bmatrix} \begin{bmatrix} v^k \\ y^k \end{bmatrix} \tag{4.5}$$

$$\lambda = \frac{w + 1 - \phi \pm \sqrt{(w + 1 - \phi)^2 - 4w}}{2} \tag{4.6}$$

連続型 PSO アルゴリズム (Continuous Particle Swarm Optimization; CPSO) について述べる．

ベクトル y と $\mathrm{sgn}(y)$ の要素によって与えられる対角要素を持つ対角行列を $\mathrm{diag}[y]$ とする．y の σ 関数を表す．として $\mathrm{sgn}(y) = 1$ if $y > 0$ の場合は, $\mathrm{sgn}(y) = -1$ if $y < 0$.

したがって，正の定数であると仮定すると，最小化のために X の進化を近似することが提案される． また CPSO の安定性解析も議論されている [41].

状態変数 X，V，X_{db} はベクトルではなく，以前に定義された適切な次元の行列であるため、上記の表記法は標準状態空間表記法ではない． また以下に CPSO の位置と速度の更新式 (式 (4.7), 式 (4.8)) と，アルゴリズムについて示す．問題解決の実行可能領域を考え，行列による連続時間 PSO 動力学を示す (式 (4.9), 式 (4.10), 式 (4.11)).

CPSO アルゴリズム

- 1.X, V とパラメータ α, β, γ と a の初期値を設定する．
- 2.X_{db}, X_{gb} の初期値を導出する．
- 3.$\dot{V}$ を計算して,V を更新する．
- 4.X を更新して X_{db}, X_{gb} を評価する．
- 5. 収束すると仮定した場合は終了しそれ以外の場合は 3 から繰り返す．

$$\dot{X} = V \tag{4.7}$$
$$\dot{V} = -\alpha V + \beta(X_{db} - X) + \gamma(X_{gb}T - X) \tag{4.8}$$
$$\dot{X_{db}} = a(X - X_{db})[I_n + diag[sgn(F(X_{gb} - F(X)))]] \tag{4.9}$$
$$\dot{X_{gb}} = X_{db}Q_j \tag{4.10}$$
$$j = arg \inf_{0<i\leq n}(f(x_{db})) \tag{4.11}$$

オリジナルの PSO アルゴリズムに含まれる恣意性を少なくし，より効率的かつ高精度な探索を実現するために勾配法による速度評価を導入している [42] [43]. 運動性素子が自分の置かれた近くの環境を知覚してより適合度の高い空間座標を獲得するために，以下のようなセンサリング・アルゴリズムを搭載する.

勾配によるスケーリングパラメータの導入を行う．粒子が投入された探索空間 (ξ, η) には問題に応じた 目的関数 Q が定義されており，粒子はその最大値か最小値を探索するものとする. 現時間ステップ k における粒子の位置座標を $(\xi k, \eta k)$ とし，その座標における目的関数の値を Q_k として，粒子の移動に伴う目的関数の変化に注目すると次のような目的関数の離散的な勾配 α が得られる.

勾配 α を，$v_i^{k+1} = \beta^k v_i^k$，$\beta^k = \alpha^{k-1/\alpha k}$ と置くことでランドスケープに合わせた調整が可能となる．つまり，最適点が遠いと思うなら早く，近いと感じるならば遅く移動する．よってオリジナル PSO より精密な探索が実施できる．

本節ではCPSOに勾配情報の要素を加えた勾配PSOについて解説する．PSOの応用法であるCPSOの応用法であり，X, Y の二つの行列に加えて Z を加えかつ，いくつかのパラメータを与えて再急降下法を用いる．以下は Z を表す (式 4.12)，

勾配PSOアルゴリズム

- 1.X, V とパラメータ α, β, γ と a の初期値を設定する．
- 2.X_{db}, X_{gb}, X_{μ} から Z の初期値を導出する．
- 3.$\dot{Z}$ を計算して,Z を更新する．
- 4.$\dot{V}$ を計算して,V を更新する．
- 5.X を更新して X_{db}, X_{gb} と Z を評価する．
- 6.X_{μ} を更新する．
- 7. 収束すると仮定した場合は終了しそれ以外の場合は 3 から繰り返す．

$$Z = \beta(X_{db} - X) + \gamma(X_{gb}T - X) + \gamma(X_{\mu}) \tag{4.12}$$

以下の条件を考慮する (式 (4.13))．X_0 は初期位置行列である．

$$X = X_0 + \int_0^t V(s)ds \tag{4.13}$$

よって勾配PSOの更新式を以下に示す．ここでは X, V, X_{db}, X_{gb}, T の次元は簡略化のため省略する．

$$\begin{aligned}
\dot{X} &= V && (4.14)\\
\dot{V} &= -\alpha V + Z && (4.15)\\
\dot{Z} &= \beta(\dot{X_{db}} - \dot{X}) + (\dot{X_{gb}}T - \dot{X}) + \delta(\dot{X_{\mu}}) && (4.16)\\
\dot{X_{db}} &= a(X - X_{db})[I_n + diag[sgn(F(X_{gb} - F(X)))]] && (4.17)\\
\dot{X_{gb}} &= X_{db}Q_j, j = arg \inf_{0<i\leq n}(f(x_{db_i})) && (4.18)
\end{aligned}$$

$\alpha, \beta, \gamma, \delta$ などの実数は，PSOと勾配情報を調整するために重み付けするパラメータである． X_{μ} はニューラルネットワークのダイナミクスに由来する新しい行列である． X_{μ} は以下で定義する (式 (4.19))．

$$\begin{aligned}
x_{\mu i} &= -C\sum_{i=1} n\frac{\partial f(y_i(t))}{\partial y_i}\frac{\partial \varphi(x(t))}{\partial x_i} && (4.19)\\
\ddot{x_i} &= -\alpha\dot{x_i}(t) + z_i(t) && (4.20)\\
y_i(t) &= \varphi(x_i(t)) && (4.21)
\end{aligned}$$

したがって $z^k{}_i$ は Z のベクトルである．次に，差分法を適用する．理論的な分析の観点から, PSO の $\dot{V}$ と $\dot{x_i}$ は同等のものとみなす. $\beta(\dot{X_{db}} - \dot{X}) + \gamma(\dot{X_{gb}}T - \dot{X})$ は PSO の速度を制御する．$\delta(\dot{X_\mu})$ は勾配情報を制御する.

PSO が有するグローバル探索，ニューラルネットワークが持つ局所最急降下法などがある． 連続時間モデルでは，PSO とニューラルネットワークの組み合わせの理論的アルゴリズムが考慮されるが，分散モデルによって数値シミュレーションが行われる． サンプリング時間の設定は，係数 (β, γ, δ) の値によって変化する．したがって，X_μ を計算して取得する．

§ 4.2 制約がある場合の PSO

前節までは PSO に勾配情報を加える説明を行った．本節ではそこに制約条件を加える．連続時間 PSO アルゴリズムについて述べる．
PSO の更新式を力学系モデルとみなし，その連続化を試みると，

$$\frac{dx^p(t)}{dt} = c\int_0^t e^{-a(t-\tau)}[F^p(x^p(\tau),\tau) + C(x^p(\tau),\tau)]d\tau \tag{4.22}$$

$$\frac{d^2x^p(t)}{dt^2} + a\frac{dx^p(t)}{dt} = c[F^p(x^p(t),t) + C(x^p(t),t)] \tag{4.23}$$

またそれぞれの関数 F^p, C は以下のようになる．

$$F^p(x^p,t) = c_1(x^p(T^p(t) - x^p)) \tag{4.24}$$

$$C^p(x^p,t) = c_2(x^{Q(t)}(T^o(t) - x^p)) \tag{4.25}$$

また，2 階微分方程式で表される連続時間系モデルの状態変数表現を，$u^p(t) = x^p(t)$ ，$v^p(t) = du^p(t)/dt + au^p(t)$ とおいて導入すると，離散時間系に対応した連続系の内部状態表現モデル，

$$du^p(t)/dt = -au^p(t) + v^p(t) \tag{4.26}$$

$$dv^p(t)/dt = c[F^p(u^p(t),t) + C(u^p(t),t))] \tag{4.27}$$

を得ることができる．

次に提案手法であるハイブリッド PSO について解説する．PSO の応用法である連続時間 PSO アルゴリズムの応用法であり，勾配情報を加えることにより，精密な探索を行うことを狙いとしている．
(式 (4.22), 式 (4.23)) に勾配情報を加えるとそのモデルは，

$$\frac{dx^p(t)}{dt} = c\int_0^t e^{-a(t-\tau)}[F^p(x^p(\tau),\tau) + C(x^p(\tau),\tau) - \nabla E(x^p(\tau),\tau)]\tau \tag{4.28}$$

$$\frac{d^2x^p(t)}{dt^2} + a\frac{dx^p(t)}{dt} = c[F^p(x^p(t),t) + C(x^p(t),t) - \nabla E(x^p(t),t)] \tag{4.29}$$

またそれぞれの関数は以下のようになる．

$$F^p(x^p,t) = c_1(x^p(T^p(t)-x^p)) \tag{4.30}$$
$$C^p(x^p,t) = c_2(x^{Q(t)}(T^o(t)-x^p)) \tag{4.31}$$
$$\nabla E(x^p,t) = c_3\frac{\partial E(x^p,t)}{\partial x^p} \tag{4.32}$$

解説したままのモデルでは無制約なので，制約条件に対応したモデルである上下限制約連続時間 PSO モデルを以下の式に示す．上下限制約付最適化問題

$$min \quad E \quad (x) \tag{4.33}$$
$$subj.top_i \leq \quad x_i \quad \leq q_i, i = 1...,n \tag{4.34}$$

を直接解くために，この上下限制約領域内に問題の変数を変換して無制約化した新たな変数空間に無制約 PSO モデルを適用した「変数変換モデル」を導入する．非線形変数変換モデルを作成するために，

$$x_i = f_i(y_i) = \frac{q_i + p_i exp(-y_i)}{1+exp(-y_i)} \tag{4.35}$$

とおく．この変換式を制約条件付き問題に代入して変数 x を消去すると，

$$minE(f(y)) \tag{4.36}$$

を得ることができる．よって式 (4.29) に対応させると，

$$\frac{d^2y^p(t)}{dt^2} + a\frac{dy^p(t)}{dt} = c[F^p(y^p(t),t) + C(y^p(t),t) - \nabla E(y^p(t),t)] \tag{4.37}$$

またそれぞれの関数は以下のようになる．

$$F^p(y^p,t) = c_1(y^p(T^p(t)-y^p)) \tag{4.38}$$
$$C^p(y^p,t) = c_2(y^{Q(t)}(T^o(t)-y^p)) \tag{4.39}$$
$$\nabla E(y^p,t) = c_3\frac{\partial E(y^p,t)}{\partial y^p} \tag{4.40}$$

次にプログラムへの実装を考えた時に，連続式のままではプログラムに実装することが難しいので，オイラー法を用いて連続式を離散化し非線形変数変換モデルの離散化 PSO を作成．それぞれに対応する式を以下に示す．

$$u^p(k+1) = (1-a\triangle T)u^p(k) + \triangle T v^p(k) \tag{4.41}$$
$$v^p(k+1) = v^p(k) + c\triangle T[F^p(u^p(k),k) + C(u^p(k),k) - \nabla E(u^p(k),k)] \tag{4.42}$$
$$F^p(k,k) = c_1(u^p(l^p(k)) - u^p(k)) \tag{4.43}$$
$$C^p(k,k) = c_2(u^{Q(k)}(l^o(k)) - u^p(k)) \tag{4.44}$$
$$\nabla E(k,k) = c_3\frac{\partial E(k,t)}{\partial k} \tag{4.45}$$
$$l^p(k) = argmin(E(x^p(l))|l = 0,...,k) \tag{4.46}$$
$$(Q(k), l^o(k)) = argmin(E(x^q(l))|q = 1,2,...,P, l = 0,1,..,k) \tag{4.47}$$
$$x^p{}_i(k) = f_i(u^p{}_i(k)) = \frac{q_i + p_i exp(-u^p{}_i(k))}{1+exp(-u^p{}_i(k))}, i = i,...,n \tag{4.48}$$

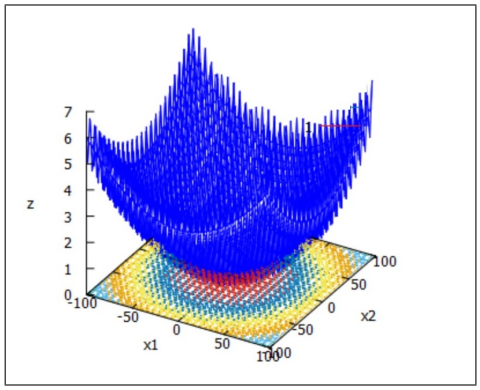

図 4.2: Griewank[30]

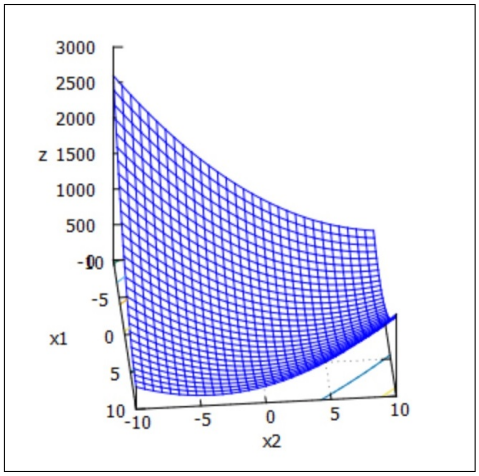

図 4.3: Booth function[31]

提案手法の有効性を示すため多峰性ならびに非常に多くの局所解をもつ評価関数として Griewank function[30](図 4.2 参照)(式 (4.49), 式 (4.50))，単峰性を持つ関数として Booth function [31](図 4.3 参照)(式 (4.51), 式 (4.52)) を用いて数値実験を行う．

$$f(x_1, ...x_n) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) \tag{4.49}$$

$$-600 \leqq x_i \leqq 600 \quad , \quad f_{min}(0, ..., 0) = 0 \tag{4.50}$$

$$f(x_1, x_2) = (x_1 + 2x_2 + -7)^2 + (2x_1 + x_2 - 5)^2 \tag{4.51}$$

$$-10 \leqq x_1, x_2 \leqq 10 \quad , \quad f_{min}(1, -3) = 0 \tag{4.52}$$

以下に従来法と提案手法を比較した物を示す (図 4.4,4.5 参照)．各パラメータの値は N（粒子の数）=10,C=1.0,c_1・c_2=1.4,c_3=0.1,a=1.0,$\Delta T = 0.9$ というように与えた．また試行回数は 100 とした．Griewank と Booth の結果を比較すると Griewank の場合では従来法と提案手法ともに最適値に収束していない．一方 Booth の結果は従来法では最適値に収束していないのに対して，提案手法では最適値に試行回数 40 以内に収束している．

よって提案手法は多峰性をもつ関数の場合にあまり有効ではないが，単峰性をもつ関数の場合には有効に働くことを示すことができた．

また Booth function の実行時に粒子の数を N=100 に変更した場合の結果 (図 4.6 参照) も示すとともに，制約条件として Booth function の実行時に $-10 < x < -5, -10 < y < -5$ の制約条件を付けくわえたものの結果 (図 4.7) も示す．この制約条件の範囲を付け加えた場合の最適値の座標は図 4.3 より $(-5, -5)$ になる．そのときの最適値は式 (4.53) より 884 となり，図 4.7 と一致し，制約条件が働いていることがわかる．

$$\begin{aligned} f(-5, -5) &= (-5 - 10 - 7)^2 + (-10 - 5 - 5)^2 \\ &= 884 \end{aligned} \tag{4.53}$$

[30]https://qiita.com/tomitomi3/items/d4318bf7afbc1c835dda griewank-function
[31]https://qiita.com/tomitomi3/items/d4318bf7afbc1c835dda booth-function

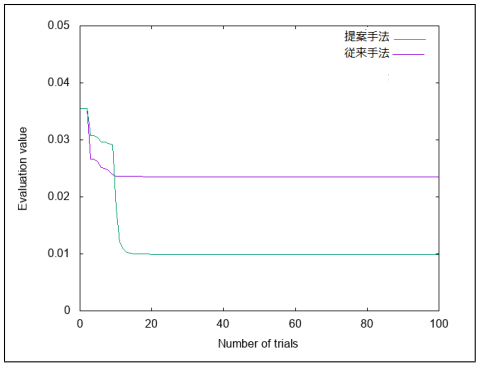

図 4.4: Griewank 実行結果

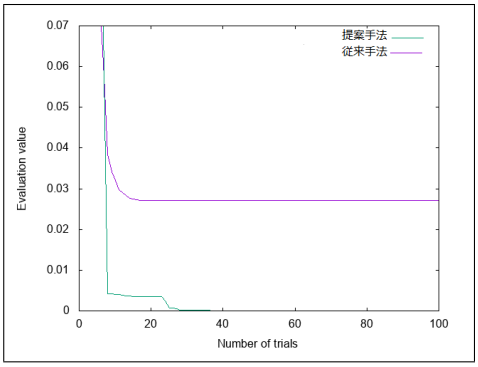

図 4.5: Booth 実行結果

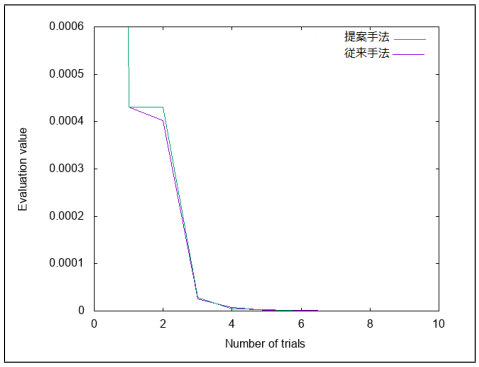

図 4.6: Booth 実行結果 N=100 の場合

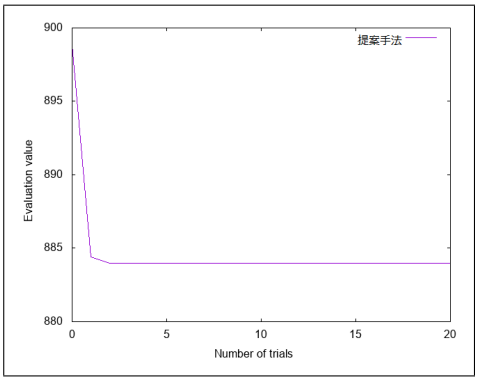

図 4.7: Booth 実行結果 制約条件付

発生させる粒子の数を 100 から 10 を比較した場合，従来法では最適値を導出できなかったが，提案手法では最適値を導出することができた．よって提案手法では従来法よりも計算コストを 10 分の 1 にしても良い結果が得られる，つまり計算コストを大幅に削減することができる．また，制約条件付きの結果では，制約条件が無しの場合と収束する値が違うことから，制約条件が働いていることが確認できる．

§ 4.3　PSO によるクラスタリング

研究により，データクラスタリングを最適化問題と見なすことが可能になった．この見解は，候補クラスタセントロイドの集合を発展させるために PSO アルゴリズムを適用する機会を提供し，それにより手近なデータセットの最適に近い分割を決定する．PSO の重要な利点は，いくつかの候補解を同時に維持し，再結合し，比較することによって局所的な最適条件に対処できることである． 対照的に，シミュレーテッドアニーリングアルゴリズム [44] のような局所探索ヒューリスティックは単一の候補解を改良するだけであり，局所最適条件に対処することにおいて悪名高いことが知られている．K 平均法などのアルゴリズムで使用される決定論的局所検索は，常に検索の開始位置から最も近い局所最適条件に収束する．

PSO ベースのクラスタリングアルゴリズムは，Omran らによって最初に導入された [45].

に記載されている．Omran らの結果 [45] [46] は，PSO ベースの方法が K-means，FCM，その他いくつかの最先端のクラスタリングアルゴリズムより優れていることを示した．Omran らの方法ではクラスタリングアルゴリズムの性能を判断するために，量子化誤差に基づく適応度を使用した．量子化誤差は次のように定義される．

$$J_e = c_3 \frac{\sum_{i=1}^{K} \sum_{\forall X_j \in C_i} d(X_j, Vi)/n_i}{K} \tag{4.54}$$

ここで，C_i は i 番目のクラスタ中心，n_i は i 番目のクラスタに属するデータ点の数である．PSO アルゴリズムの各粒子は，次のように K 個のクラスタ重心の可能なセットを表す．

$$\overrightarrow{Z}_i(t) = \overrightarrow{V}_{i,1}, \overrightarrow{V}_{i,2}, ..., \overrightarrow{V}_{i,K} \tag{4.55}$$

ここで，$V_{i,p}$ は，i 番目の粒子の p 番目のクラスター重心ベクトルを表す． 各粒子の品質は，次の適合度関数によって測定される．

$$f(Z_i, M_i) = w_1 \overline{d}_{max} + w_2(R_{max} - d_{min}(Z_i)) + w_3 J_e \tag{4.56}$$

上記の式で，R_{max} はデータセット内の最大の特徴量であり，M_i は i 番目の粒子のクラスターへのパターンの割り当てを表す行列である． 各要素 m_i，k は，パターン X_p が i 番目の粒子のクラスタ C_k に属するかどうかを示す． ユーザー定義定数 w_1，w_2，および w_3 は，異なる副目的からの寄与を評価するために使用される． 加えて，

$$\overline{d}_{max} = \underbrace{max}_{K \in 1,2....,K} \Big(\underbrace{\sum}_{\forall X_p \in C_i, K} d(X_p, V_{i,k})/n_{i,k} \Big) \tag{4.57}$$

そして，

$$d_{min}(Z_i) = \underbrace{min}_{\in p,q,p \neq q} \big(d(V_{i,p}, V_{i,q})\big) \tag{4.58}$$

上記の式は任意のペアのクラスタ間の最小ユークリッド距離である．上記において，ni, k は，粒子 i のクラスタ C_i, k に属するパターンの数である．適応度関数は多目的最適化問題であり，これはクラスタ内距離を最小化し，クラスタ間分離を最大化し，量子化誤差を減少させる．PSO クラスタリングアルゴリズムは，以下に示す．

PSO クラスタリングアルゴリズム

- 1.K 個のランダムクラスターの中心を使って各粒子を初期化する．
- 2. 最大繰り返し回数の決定
- 3. 対象を全ての粒子にする
- 4. データセット内のすべてのパターン X_p に対して行う
- 5. すべてのクラスター重心を使って X_p のユークリッド距離を計算する
- 6. X_p に最も近い重心を持つクラスターに X_p を割り当てる
- 7. フィットネス関数 $f(Z_i, M_i)$ を計算する
- 8. 各粒子のパーソナルおよびグローバルな最良の位置を見つける
- 9. PSO の式を更新する
- 10. 終了

Van der Merwe と Engelbrecht は，一般的なデータセットをクラスタリングするための K-means アルゴリズムを提案した [47]. 群の単一粒子は，K-means アルゴリズムの結果で初期化される．群れの残りの部分はランダムに初期化される．2003 年に，Xiao らは PSO と SOM との相乗作用に基づく新しいアプローチを用いて遺伝子発現データをクラスタリングした [48]． 彼らは，酵母およびラット肝細胞の遺伝子発現データに対してハイブリッド SOM-PSO アルゴリズムを適用することによって有望な結果を得た．Paterlini らは，分割に対する代表的な点評価アプローチについて，K-means，GA [49]PSO および微分進化（DE）[50] の性能を比較した．クラスタリング結果は PSO と DE が K 平均アルゴリズムより優れていることを示した [51].

Cui らは，テキスト文書を分類するための PSO ベースのハイブリッドアルゴリズムを提案した [52]．彼らは PSO，K 平均，そして 4 つの異なるテキスト文書データセット上のハイブリッド PSO クラスタリングアルゴリズム．結果はハイブリッド PSO アルゴリズムがよりコンパクトを生成できることを示した．クラスタリングは、K-means アルゴリズムよりも短時間で行われる．

§ 4.4　提案手法のアルゴリズム

センサで収集した環境・生体データに対して PSO によるクラスタリングを行うためのアルゴリズムを以下に示す．またその全体のフローを以下に示す図 (4.8 参照).

- Step1:　センサとマイコンを組み合わせて制作した環境・生体データ収集のためのセンサ類を，被験者に装着してもらいデータを収集する．収集するデータは 20 種類のセンサデータにタイムスタンプをつけておく．

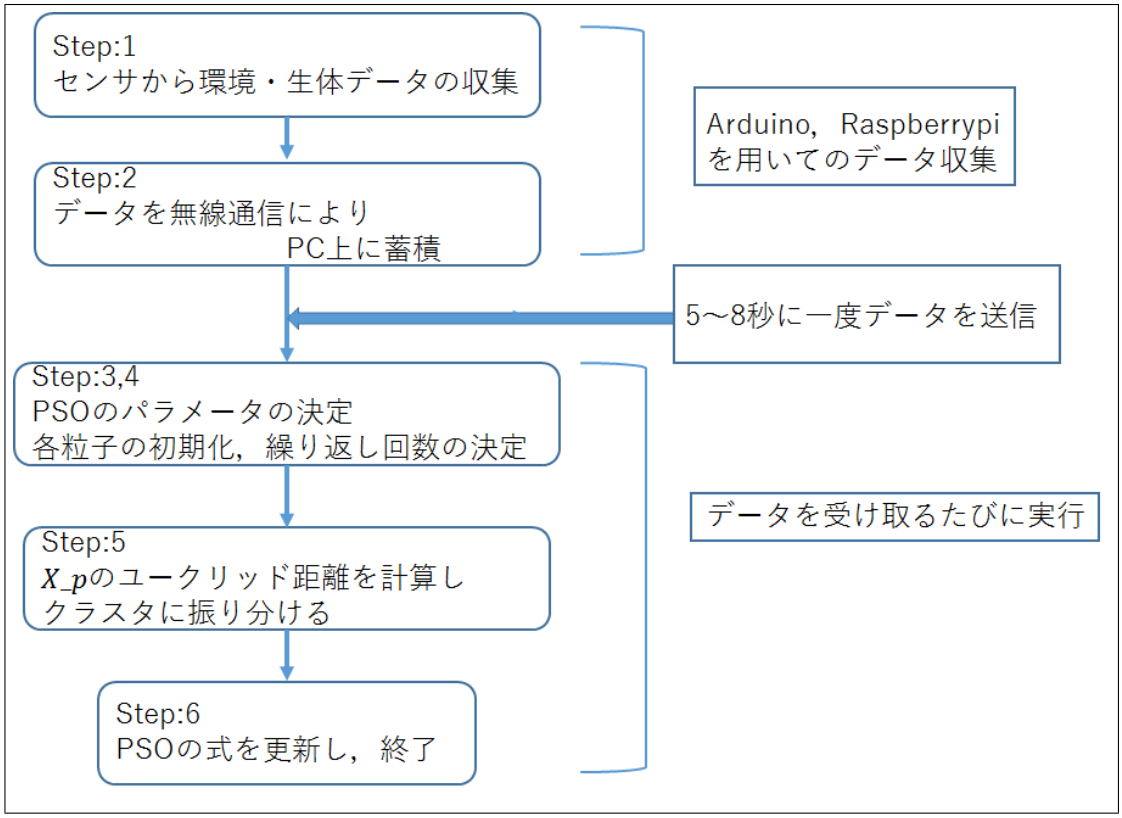

図 4.8: 提案手法の全体のフロー

- Step2:　データを無線通信により PC 上に蓄積する．その収集したデータを csv 形式のファイルで保存する．またデータは通信の遅延の関係により，5 から 7 秒程度に一度送信される．

  ここで用いるセンサ類は 2 章で述べたものである．また無線通信に関しても同様に 2 章で述べた手法を用いて行う．

- Step3:　csv 形式でデータを受け取るたびに，提案した勾配・制約を考慮した PSO によるクラスタリングを行う．そのためパラメーター $a, \Delta T, c, c_1, c_2, c_3$ を決定する．

- Step:4　送信されてきたデータに対して PSO のクラスタリングアルゴリズムにもとづきクラスタリングを行う．K 個のランダムクラスターの中心から各粒子を初期化し，最大繰り返し回数を決定する．

- Step:5　全ての粒子を対象に X_p のユークリッド距離を計算し，X_p に最も近い重心を持つクラスターに X_p を割り当てる．

- Step:6　各粒子のパーソナルおよびグローバルな最良の位置を見つけ，PSO の式を更新する．

第5章

おわりに

本研究の目的は，まず環境・生体データを収集する機器の開発として，無線通信での自動でのデータの蓄積を行うことのできるシステムを開発し，収集したデータから類似性やイベント性の検出を行い，実際に開発したシステムを用いて，いくつかの解析を行い，PSOによるクラスタリングのアルゴリズムを提案することである．

結論として，環境・生体データの収集機器の開発を行うことができた．無線通信でのデータの自動収集のシステムを作成することはできたが，センサのノイズや利便性の問題など改善が必要な点が残る結果となった．また，収集したデータを用いたKH Coderの解析に関してはあまり良い結果を得ることができず，行動識別を満足に行え無かった．しかしExcelを用いての行動分類はそれぞれの環境での行動を良く分類することができた．勾配・制約を考慮したPSOの定式化を行い，簡単な数値実験により従来のPSOとの結果から，その有効性を示すことができた．またPSOのクラスタリングアルゴリズムを提案することができた．

今後の課題としては，開発したデータの収集機器に関して，実際にユーザーが使うことを考えて利便性について改善する必要があるとともに，収集するデータのノイズの削減や，測定するデータの値を安定させることがデバイス面での課題となっている．またデータ収集の際の無線通信の遅延が起きるケースがあるので遅延の削減に取り組む必要がある．また，収集したデータに対して，提案したアルゴリズムを用いたクラスタリングを行い，従来法との有効性の比較・検証を行う必要がある．

謝辞

本研究を遂行するにあたり，多大なご指導と終始懇切丁寧なご鞭撻を賜った富山県立大学電子・情報工学科の奥原浩之教授に深甚な謝意を表します．最後になりましたが，多大な協力をして頂いた研究室の同輩諸氏に感謝致します．

2018 年 2 月

山本 聖也

参考文献

[1] 中西泰人，辻貴孝，大山実，箱崎勝也 “Context Aware Messaging Service：位置情報とスケジュール情報を用いたコミュニケーションシステムの構築及び運用実験”, 情報処理学会論文誌，Vol. 42，No. 7，pp.1847–1857，2001

[2] 相澤清晴, “ライフログ”, 映像情報メディア学会誌, Vol. 63, No. 4, pp. 445–448, 2009.

[3] 三井紀子，酒井豊子，中島利誠 “運動中の衣服下気候と着心地に及ぼす繊維の影響”, 日生気誌，Vol. 23，No. 1，pp.35–42，1986

[4] 柳平雅俊，安士光男 “運転状態推定技術の開発— 心拍解析による眠気状態の検出—”, PIONEER R & D，Vol. 14，No. 3，pp.17–27，2003

[5] 前中一介 “絆創膏型人体活動モニタリングシステム”, 人工臓器，Vol. 42，No. 1，pp.17–27，2013

[6] 芳竹宣裕, 伊藤愼, “ユビキタス環境が生み出す大量情報 「ライフログ」 の活用と実装技術”, NEC 技報, Vol. 62, No. 4, p. 77, 2009.

[7] 新保史生, “ライフログの定義と法的責任 個人の行動履歴を営利目的で利用することの妥当性”, 情報管理, Vol. 53, No. 6, pp. 295–310, 2010.

[8] 角田宏貴, Hiroki SUMIDA, “ライフログ分析による行動特徴抽出及びイベント検出”, 法政大学大学院紀要（情報科学研究科編）, Vol. 9, pp. 119–124, 2014.

[9] 矢野裕司, 横井健, 橋山智訓, “行動辞書を利用した Twitter からの行動抽出”, 情報科学技術フォーラム講演論文集, Vol. 11, No. 4, pp. 51–56, 2012.

[10] 緒方広明, “日本語学習を支援するユビキタス学習環境に関する研究”, http://www.taf.or.jp/files/items/542/File/P212.pdf, 閲覧日 2019,1,30.

[11] 啓之田中, “位置情報の規律のあり方：スマートフォン時代の利便性とプライバシー”, 人間社会研究, Vol. 11, pp. 75–85, 2014.

[12] 中村匡秀, 下條彰, 井垣宏, “異なるライフログを集約するための標準データモデルの考察 ”, 電子情報通信学会技術研究報告, Vol. 109, No. 272, pp. 1847–1857, 2001.

[13] 北村圭吾, 山崎俊彦, 相澤清晴, “食事ログの取得と処理－画像処理による食事記録－”, 映像情報メディア学会誌, Vol. 63, No. 3, pp. 376–379, 2009.

[14] 株式会社ＮＴＴデータ経営研究所, “日本語学習を支援するユビキタス学習環境に関する研究”, http://www.keieiken.co.jp/aboutus/newsrelease/161122/, 閲覧日 2018,2,5.

[15] 寺田努, “特集●インタラクティブシステムとソフトウェア”, コンピュータソフトウェア, Vol. 28, No. 2, pp. 43–54, 2011.

[16] 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 Vol. 54, No. 3, pp. 15–22, 2006.

[17] 品川住満, 谷川智宏, 太田茂, “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, 1999.

[18] 池浦良淳, 大塚英樹, 猪岡光, “皮膚電気反射に基づくロボット運動の心理的評価に関する考察”, 人間工学 Vol. 31, No. 5, pp. 355–358, 1995.

[19] 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・共有方式-モバイルコンピューティング, モバイルアプリケーション, ユビキタス通信, モバイルマルチメディア通信-”, 電子情報通信学会技術研究報告. MoMuC, モバイルマルチメディア通信, Vol. 108, No. 44, pp. 115–120, 2008.

[20] 寺田努, “ウェアラブルセンサを用いた行動認識技術の現状と課題”, コンピュータ ソフトウェア, Vol. 28, No. 2, pp. 43–54, 2011.

[21] 貴志一樹, 山崎俊彦, 相澤清晴, “机上行動のライフログのための行動認識”, 映像情報メディア学会年次大会講演予稿集, Vol. 2014, , 2014.

[22] 佐藤誠, 森田千絵, 土井美和子 “生体と加速度データを用いた行動認識’, 情報処理学会全国大会講演論文集, Vol. 65, pp. 239–242, 2003.

[23] 樋口耕一, “テキスト型データの計量的分析 : 2 つのアプローチの峻別と統合”, 理論と方法, Vol. 19, No. 1, pp. 101–115, 2004.

[24] 佐野香織, 李在鎬, “KH Coder で何ができるか : 日本語習得・日本語教育研究利用への示唆”, 言語文化と日本語教育, Vol. 33, pp. 94–95, 2007.

[25] “KH Coder を用いた研究事例のリスト”, http://khc.sourceforge.net/bib.html, 閲覧日 2019,1,7.

[26] “クラスター分析の手法（階層クラスター分析） — データ分析基礎知識”, https://www.albert2005.co.jp/knowledge/data_mining/cluster/hierarchical_clustering, 閲覧日 2019,1,24.

[27] “階層的クラスター分析について”, http://koichi.nihon.to/cgi-bin/bbs_khn/khcf.cgi?list=&no=977&mode=allread&page=0, 閲覧日 2019,1,24.

[28] 吉原一紘, 徳高平蔵, “クラスター分析の概要”, *Journal of Surface Analysis*, Vol. 21, No. 1, pp. 10–17, 2014.

[29] 李美龍, 田中恒也, 成田吉弘, “画像を用いた製品の「飽き」に関する感性評価: －デザインの視覚的要素を中心に－”, 日本感性工学会論文誌, Vol. 11, No. 3, pp. 407–417, 2012.

[30] 小峯敦・下平裕之, “ベヴァリッジ『自由社会における完全雇用』のケインズ的要素-テキストマイニングを加味した量的・質的分析-”, 2017.

[31] 齋藤堯幸, “多次元尺度構成法”, 計測と制御, Vol. 22, No. 1, pp. 126–131, 1983.

[32] Joseph B Kruskal, “Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis”, *Psychometrika*, Vol. 29, No. 1, pp. 1–27, 1964.

[33] “Jaccard 係数の計算式と特徴（1）”, https://www.slideshare.net/khcoder/jaccard1, 閲覧日 2018,2,3.

[34] 中山慶一郎, “< 研究ノート > 対応分析によるデータ解析”, 関西学院大学社会学部紀要, No. 108, pp. 133–145, 2009.

[35] T. KOHONEN, “Self-organized formation of topologically correct feature map”, *Biol. Cybern.*, Vol. 43, pp. 59–69, 1982.

[36] 岡晋之介, “自己組織化マップを用いた気象要素の分類と予測”, http://www.gifu-nct.ac.jp/elec/deguchi/sotsuron/oka/oka.html, 閲覧日 2019,1,7.

[37] “自己組織化特徴マップ（SOM）”, http://www.sist.ac.jp/ kanakubo/research/neuro/selforganizingmap.html, 閲覧日 2019,1,31.

[38] J. Kennedy, R. C. Eberhart, “Particle swarm optimization”, $IEEEConf.OnNeuralNetworks, IV, Piscataway, NJ$, pp. 1942–1948. 1995.

[39] J. Kennedy, R. C. Eberhart, Y. Shi, “Swarm intelligence”, Morgan Kaumann Publishers, San Fran-cisco, CA, pp. 1942–1948, 2001.

[40] 石亀　敦司, 安田　恵一郎, “群れの知能： Particle Swarm Optimization”, 知能と情報（日本知能情報ファジィ学会誌）, Vol. 20, No. 6 , pp. 829–839, 2008.

[41] H. M. Emara, H. A. Abdel Fattah, “Continuous swarm optimization technique with stability analysis”, $Proceedingofthe2004AmericanControlConference$, 2811–2817, 2004.

[42] Ryuzaburo Sugino, “Numerical Performance of PSO Algorithm Using the Gradient Method”, $InformationProcessingLetters$, Vol. 57, pp. 461–468, 2009.

[43] M. Jiang, Y. P. Luo and S. Y. Yang, “Stochastic convergenceanalysis and parameter selection of the standard particle swarm optimization algorithm”, $InformationProcessingLetters$, Vol. 102, No. 1, pp. 8–16, 2007.

[44] Selim SZ, Alsultan K, “A simulated annealing algorithm for the clustering problem”, Pattern recognition, 24(7), pp. 1003–1008, 1991.

[45] Omran M, Salman A, Engelbrecht AP, “Image Classification using Particle Swarm Optimization”, In Conference on Simulated Evolution and Learning, Vol. 1, pp. 370–374, 2002.

[46] Omran M, Engelbrecht AP, Salman A, "Particle Swarm Optimization Method for Image Clustering", International Journal of Pattern Recognition and Artificial Intelligence, 19(3), pp. 297–322, 2005.

[47] van der Merwe DW, Engelbrecht AP, "Data clustering using particle swarm optimization", In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation, pp. 215–220, 2003.

[48] Xiao X, Dow ER, Eberhart RC, Miled ZB and Oppelt RJ, "Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization", Proc of the 17th International Symposium on Parallel and Distributed Processing (PDPS'03), IEEE Computer Society, Washington DC. 2003.

[49] Holland JH, "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor. 1975.

[50] Storn R, Price K, "Differential evolution A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", $ComputationalStatisticsandDataAnalysis$, Vol. 50, pp. 341–359, 1997.

[51] Paterlini S, Krink T, "Differential Evolution and Particle Swarm Optimization in Partitional Clustering", $ComputationalStatisticsandDataAnalysis$, Vol. 50, pp. 1220–1247, 2006.

[52] Cui X, Potok TE, "Document Clustering Analysis Based on Hybrid PSO+Kmeans Algorithm", Journal of Computer Sciences (Special Issue), ISSN pp. 1549–3636, 2005.

付録

A. 1　環境・生体データ収集アプリケーションのソースコード

ライフログデータ取得アプリケーションの Arduino1 のソースコード 1 をしめす.

ソースコード A. 1: app1.ino

```
1  //
       /=============================================================
2  //
       /=============================================================
3  //
       /=============================================================

4
5  //============================
6  ////////// Temperature //////////
7  //============================
8
9  #include <Wire.h>
10
11 #define BME280_ADDRESS 0x76
12 unsigned long int hum_raw,temp_raw,pres_raw;
13 signed long int t_fine;
14
15 uint16_t dig_T1;
16  int16_t dig_T2;
17  int16_t dig_T3;
18 uint16_t dig_P1;
19  int16_t dig_P2;
20  int16_t dig_P3;
21  int16_t dig_P4;
22  int16_t dig_P5;
23  int16_t dig_P6;
24  int16_t dig_P7;
25  int16_t dig_P8;
26  int16_t dig_P9;
27  int8_t dig_H1;
28  int16_t dig_H2;
29  int8_t dig_H3;
30  int16_t dig_H4;
31  int16_t dig_H5;
32  int8_t dig_H6;
33
34 //
       /=============================================================
35 void readTrim()
36 {
37     uint8_t data[32],i=0;
38     Wire.beginTransmission(BME280_ADDRESS);
39     Wire.write(0x88);
```

```
40      Wire.endTransmission();
41      Wire.requestFrom(BME280_ADDRESS,24);
42      while(Wire.available()){
43          data[i] = Wire.read();
44          i++;
45      }
46
47      Wire.beginTransmission(BME280_ADDRESS);
48      Wire.write(0xA1);
49      Wire.endTransmission();
50      Wire.requestFrom(BME280_ADDRESS,1);
51      data[i] = Wire.read();
52      i++;
53
54      Wire.beginTransmission(BME280_ADDRESS);
55      Wire.write(0xE1);
56      Wire.endTransmission();
57      Wire.requestFrom(BME280_ADDRESS,7);
58      while(Wire.available()){
59          data[i] = Wire.read();
60          i++;
61      }
62      dig_T1 = (data[1] << 8) | data[0];
63      dig_T2 = (data[3] << 8) | data[2];
64      dig_T3 = (data[5] << 8) | data[4];
65      dig_P1 = (data[7] << 8) | data[6];
66      dig_P2 = (data[9] << 8) | data[8];
67      dig_P3 = (data[11]<< 8) | data[10];
68      dig_P4 = (data[13]<< 8) | data[12];
69      dig_P5 = (data[15]<< 8) | data[14];
70      dig_P6 = (data[17]<< 8) | data[16];
71      dig_P7 = (data[19]<< 8) | data[18];
72      dig_P8 = (data[21]<< 8) | data[20];
73      dig_P9 = (data[23]<< 8) | data[22];
74      dig_H1 = data[24];
75      dig_H2 = (data[26]<< 8) | data[25];
76      dig_H3 = data[27];
77      dig_H4 = (data[28]<< 4) | (0x0F & data[29]);
78      dig_H5 = (data[30] << 4) | ((data[29] >> 4) & 0x0F);
79      dig_H6 = data[31];
80  }
81
82  //      /============================================================
83  void writeReg(uint8_t reg_address, uint8_t data)
84  {
85      Wire.beginTransmission(BME280_ADDRESS);
86      Wire.write(reg_address);
87      Wire.write(data);
88      Wire.endTransmission();
89  }
90
91  //      /============================================================
92  void readData()
93  {
94      int i = 0;
95      uint32_t data[8];
96      Wire.beginTransmission(BME280_ADDRESS);
```

```
97      Wire.write(0xF7);
98      Wire.endTransmission();
99      Wire.requestFrom(BME280_ADDRESS,8);
100     while(Wire.available()){
101         data[i] = Wire.read();
102         i++;
103     }
104     pres_raw = (data[0] << 12) | (data[1] << 4) | (data[2] >> 4);
105     temp_raw = (data[3] << 12) | (data[4] << 4) | (data[5] >> 4);
106     hum_raw = (data[6] << 8) | data[7];
107 }
108
109 //      /=========================================================
110 signed long int calibration_T(signed long int adc_T)
111 {
112
113     signed long int var1, var2, T;
114     var1 = ((((adc_T >> 3) - ((signed long int)dig_T1<<1))) * ((signed long int)
            dig_T2)) >> 11;
115     var2 = (((((adc_T >> 4) - ((signed long int)dig_T1)) * ((adc_T>>4) - ((signed
            long int)dig_T1))) >> 12) * ((signed long int)dig_T3)) >> 14;
116
117     t_fine = var1 + var2;
118     T = (t_fine * 5 + 128) >> 8;
119     return T;
120 }
121
122 //      /=========================================================
123 unsigned long int calibration_P(signed long int adc_P)
124 {
125     signed long int var1, var2;
126     unsigned long int P;
127     var1 = (((signed long int)t_fine)>>1) - (signed long int)64000;
128     var2 = (((var1>>2) * (var1>>2)) >> 11) * ((signed long int)dig_P6);
129     var2 = var2 + ((var1*((signed long int)dig_P5))<<1);
130     var2 = (var2>>2)+(((signed long int)dig_P4)<<16);
131     var1 = (((dig_P3 * (((var1>>2)*(var1>>2)) >> 13)) >>3) + ((((signed long int)
            dig_P2) * var1)>>1))>>18;
132     var1 = ((((32768+var1))*((signed long int)dig_P1))>>15);
133     if (var1 == 0)
134     {
135         return 0;
136     }
137     P = (((unsigned long int)(((signed long int)1048576)-adc_P)-(var2>>12)))*3125;
138     if(P<0x80000000)
139     {
140        P = (P << 1) / ((unsigned long int) var1);
141     }
142     else
143     {
144         P = (P / (unsigned long int)var1) * 2;
145     }
146     var1 = (((signed long int)dig_P9) * ((signed long int)(((P>>3) * (P>>3))>>13)))
            >>12;
147     var2 = (((signed long int)(P>>2)) * ((signed long int)dig_P8))>>13;
148     P = (unsigned long int)((signed long int)P + ((var1 + var2 + dig_P7) >> 4));
149     return P;
```

```
150 }
151
152 //
      /=============================================================
153 unsigned long int calibration_H(signed long int adc_H)
154 {
155     signed long int v_x1;
156
157     v_x1 = (t_fine - ((signed long int)76800));
158     v_x1 = (((((adc_H << 14) -(((signed long int)dig_H4) << 20) - (((signed long int
          )dig_H5) * v_x1)) +
159                 ((signed long int)16384)) >> 15) * (((((((v_x1 * ((signed long int)dig_H6
                      )) >> 10) *
160                 (((v_x1 * ((signed long int)dig_H3)) >> 11) + ((signed long int) 32768)))
                        >> 10) + (( signed long int)2097152)) *
161                 ((signed long int) dig_H2) + 8192) >> 14));
162    v_x1 = (v_x1 - (((((v_x1 >> 15) * (v_x1 >> 15)) >> 7) * ((signed long int)dig_H1))
          >> 4));
163    v_x1 = (v_x1 < 0 ? 0 : v_x1);
164    v_x1 = (v_x1 > 419430400 ? 419430400 : v_x1);
165    return (unsigned long int)(v_x1 >> 12);
166 }
167
168 ////////// Hikari //////////
169
170 int val = 0;
171 int iPin = A3;
172 int oPin = 13;
173
174 //============================
175 ////////// Jinkan ///////
176 //============================
177
178 const int ledPin = 13;
179 const int pirPin = 2;
180
181 //============================
182 ////////// 9jiku ////////
183 //============================
184
185 #include<Wire.h>
186
187 #define Addr_Accl 0x19
188
189 #define Addr_Gyro 0x69
190
191 #define Addr_Mag 0x13
192
193
194 float xAccl = 0.00;
195 float yAccl = 0.00;
196 float zAccl = 0.00;
197 float xGyro = 0.00;
198 float yGyro = 0.00;
199 float zGyro = 0.00;
200 int xMag = 0;
201 int yMag = 0;
202 int zMag = 0;
203
204 //
```

```
/=============================================================
void BMX055_Init()
{
  //
      ------------------------------------------------------------
  Wire.beginTransmission(Addr_Accl);
  Wire.write(0x0F); // Select PMU_Range register
  Wire.write(0x03); // Range = +/- 2g
  Wire.endTransmission();
  //delay(100);
  //
      ------------------------------------------------------------
  Wire.beginTransmission(Addr_Accl);
  Wire.write(0x10); // Select PMU_BW register
  Wire.write(0x08); // Bandwidth = 7.81 Hz
  Wire.endTransmission();
  //delay(100);
  //
      ------------------------------------------------------------
  Wire.beginTransmission(Addr_Accl);
  Wire.write(0x11); // Select PMU_LPW register
  Wire.write(0x00); // Normal mode, Sleep duration = 0.5ms
  Wire.endTransmission();
  //delay(100);
  //
      ------------------------------------------------------------
  Wire.beginTransmission(Addr_Gyro);
  Wire.write(0x0F); // Select Range register
  Wire.write(0x04); // Full scale = +/- 125 degree/s
  Wire.endTransmission();
  //delay(100);
  //
      ------------------------------------------------------------
  Wire.beginTransmission(Addr_Gyro);
  Wire.write(0x10); // Select Bandwidth register
  Wire.write(0x07); // ODR = 100 Hz
  Wire.endTransmission();
  //delay(100);
  //
      ------------------------------------------------------------
  Wire.beginTransmission(Addr_Gyro);
  Wire.write(0x11); // Select LPM1 register
  Wire.write(0x00); // Normal mode, Sleep duration = 2ms
  Wire.endTransmission();
  //delay(100);
  //
      ------------------------------------------------------------
  Wire.beginTransmission(Addr_Mag);
  Wire.write(0x4B); // Select Mag register
  Wire.write(0x83); // Soft reset
  Wire.endTransmission();
  //delay(100);
  //
```

```
  //------------------------------------------------------------------------

  Wire.beginTransmission(Addr_Mag);
  Wire.write(0x4B); // Select Mag register
  Wire.write(0x01); // Soft reset
  Wire.endTransmission();
  //delay(100);
  //
  //------------------------------------------------------------------------

  Wire.beginTransmission(Addr_Mag);
  Wire.write(0x4C); // Select Mag register
  Wire.write(0x00); // Normal Mode, ODR = 10 Hz
  Wire.endTransmission();
 //
  //------------------------------------------------------------------------

  Wire.beginTransmission(Addr_Mag);
  Wire.write(0x4E); // Select Mag register
  Wire.write(0x84); // X, Y, Z-Axis enabled
  Wire.endTransmission();
 //
  //------------------------------------------------------------------------

  Wire.beginTransmission(Addr_Mag);
  Wire.write(0x51); // Select Mag register
  Wire.write(0x04); // No. of Repetitions for X-Y Axis = 9
  Wire.endTransmission();
 //
  //------------------------------------------------------------------------

  Wire.beginTransmission(Addr_Mag);
  Wire.write(0x52); // Select Mag register
  Wire.write(0x16); // No. of Repetitions for Z-Axis = 15
  Wire.endTransmission();
}
//
  /*=========================================================================

void BMX055_Accl()
{
  int data[6];
  for (int i = 0; i < 6; i++)
  {
    Wire.beginTransmission(Addr_Accl);
    Wire.write((2 + i));// Select data register
    Wire.endTransmission();
    Wire.requestFrom(Addr_Accl, 1);// Request 1 byte of data
    // Read 6 bytes of data
    // xAccl lsb, xAccl msb, yAccl lsb, yAccl msb, zAccl lsb, zAccl msb
    if (Wire.available() == 1)
      data[i] = Wire.read();
  }
  // Convert the data to 12-bits
  xAccl = ((data[1] * 256) + (data[0] & 0xF0)) / 16;
  if (xAccl > 2047) xAccl -= 4096;
  yAccl = ((data[3] * 256) + (data[2] & 0xF0)) / 16;
  if (yAccl > 2047) yAccl -= 4096;
  zAccl = ((data[5] * 256) + (data[4] & 0xF0)) / 16;
  if (zAccl > 2047) zAccl -= 4096;
  xAccl = xAccl * 0.0098; // renge +-2g
```

```
299   yAccl = yAccl * 0.0098; // renge +-2g
300   zAccl = zAccl * 0.0098; // renge +-2g
301 }
302 //
        /============================================================
303 void BMX055_Gyro()
304 {
305   int data[6];
306   for (int i = 0; i < 6; i++)
307   {
308     Wire.beginTransmission(Addr_Gyro);
309     Wire.write((2 + i)); // Select data register
310     Wire.endTransmission();
311     Wire.requestFrom(Addr_Gyro, 1); // Request 1 byte of data
312     // Read 6 bytes of data
313     // xGyro lsb, xGyro msb, yGyro lsb, yGyro msb, zGyro lsb, zGyro msb
314     if (Wire.available() == 1)
315       data[i] = Wire.read();
316   }
317   // Convert the data
318   xGyro = (data[1] * 256) + data[0];
319   if (xGyro > 32767) xGyro -= 65536;
320   yGyro = (data[3] * 256) + data[2];
321   if (yGyro > 32767) yGyro -= 65536;
322   zGyro = (data[5] * 256) + data[4];
323   if (zGyro > 32767) zGyro -= 65536;
324
325   xGyro = xGyro * 0.0038; // Full scale = +/- 125 degree/s
326   yGyro = yGyro * 0.0038; // Full scale = +/- 125 degree/s
327   zGyro = zGyro * 0.0038; // Full scale = +/- 125 degree/s
328 }
329 //
        /============================================================
330 void BMX055_Mag()
331 {
332   int data[8];
333   for (int i = 0; i < 8; i++)
334   {
335     Wire.beginTransmission(Addr_Mag);
336     Wire.write((0x42 + i)); // Select data register
337     Wire.endTransmission();
338     Wire.requestFrom(Addr_Mag, 1); // Request 1 byte of data
339     // Read 6 bytes of data
340     // xMag lsb, xMag msb, yMag lsb, yMag msb, zMag lsb, zMag msb
341     if (Wire.available() == 1)
342       data[i] = Wire.read();
343   }
344   // Convert the data
345   xMag = ((data[1] <<8) | (data[0]>>3));
346   if (xMag > 4095) xMag -= 8192;
347   yMag = ((data[3] <<8) | (data[2]>>3));
348   if (yMag > 4095) yMag -= 8192;
349   zMag = ((data[5] <<8) | (data[4]>>3));
350   if (zMag > 16383) zMag -= 32768;
351 }
352
353 //==========================
354 ///////// Taion ////////
355 //==========================
```

```
356
357 #include <math.h>
358
359 int analog = 1;
360 int V_0 = 3276;
361 double V_R0 =1;
362 int circuit_R0 =10000;
363
364 int thermo_B = 3380;
365 int thermo_R0 = 10000;
366 int thermo_T0 = 298;
367
368 double thermo_R = 1;
369 double temp = 25.00;
370
371 //===========================
372 /////// Shinpaku ////////
373 //===========================
374
375
376
377 int pulsePin = 0; // Pulse Sensor purple wire connected to analog pin 0
378
379 // Volatile Variables, used in the interrupt service routine!
380 volatile int BPM; // int that holds raw Analog in 0. updated every 2mS
381 volatile int Signal; // holds the incoming raw data
382 volatile int IBI = 600; // int that holds the time interval between beats! Must be seeded!
383 volatile boolean Pulse = false; // "True" when User's live heartbeat is detected. "False"
        when not a "live beat".
384 volatile boolean QS = false; // becomes true when Arduoino finds a beat.
385
386 //
      /=============================================================
387 // Sends Data to Pulse Sensor Processing App, Native Mac App, or Third-party Serial
      Readers.
388 void sendDataToSerial(char symbol, int data ){
389     //Serial.print(symbol);
390     Serial.print(data);
391 }
392
393 //===========================
394 ////////// GSR /////////
395 //===========================
396 const int GSR=A2;
397 int sensorValue=0;
398 int gsr_average=0;
399
400 //
      /=============================================================
401 //
      /=============================================================
402 //
      /=============================================================
403
404 void setup() {
405   // put your setup code here, to run once:
406
```

```
407   //////////// GPS /////////
408 // mySerial.begin(9600);
409   Serial.begin(115200);
410
411   //////////// Temperature /////////
412
413   uint8_t osrs_t = 1; //Temperature oversampling x 1
414   uint8_t osrs_p = 1; //Pressure oversampling x 1
415   uint8_t osrs_h = 1; //Humidity oversampling x 1
416   uint8_t mode = 3; //Normal mode
417   uint8_t t_sb = 5; //Tstandby 1000ms
418   uint8_t filter = 0; //Filter off
419   uint8_t spi3w_en = 0; //3-wire SPI Disable
420
421   uint8_t ctrl_meas_reg = (osrs_t << 5) | (osrs_p << 2) | mode;
422   uint8_t config_reg = (t_sb << 5) | (filter << 2) | spi3w_en;
423   uint8_t ctrl_hum_reg = osrs_h;
424
425   //Serial.begin(9600);
426   Wire.begin();
427
428   writeReg(0xF2,ctrl_hum_reg);
429   writeReg(0xF4,ctrl_meas_reg);
430   writeReg(0xF5,config_reg);
431   readTrim();
432
433   //============================
434   //////////// Hikari ///////
435   //============================
436
437   //Serial.begin(9600);
438   pinMode(oPin, OUTPUT);
439
440   //============================
441   ///////// Jinkan ////////
442   //============================
443
444   // put your setup code here, to run once:
445   //Serial.begin(9600);
446   pinMode(ledPin, OUTPUT);
447
448   //============================
449   //////////// 9jiku ////////
450   //============================
451   Wire.begin();
452   Serial.begin(115200);
453   BMX055_Init();
454
455   //============================
456   ///////// Taion ////////
457   //============================
458   //Serial.begin(9600);
459   ////Serial.println("READ_START");
460
461   //============================
462   /////// Shinpaku ////////
463   //============================
464   //Serial.begin(115200); // we agree to talk
465   interruptSetup(); // sets up to read Pulse Sensor signal every 2mS
466   // IF YOU ARE POWERING The Pulse Sensor AT VOLTAGE LESS THAN THE
          BOARD VOLTAGE,
```

```
467   // UN-COMMENT THE NEXT LINE AND APPLY THAT VOLTAGE TO THE A-
          REF PIN
468   // analogReference(EXTERNAL);
469
470   //============================
471   ////////// GSR //////////
472   //============================
473   // Serial.begin(9600);
474
475 }
476
477 //
      /=============================================================
478 //
      /=============================================================
479 //
      /=============================================================
480
481 void loop() {
482   // put your main code here, to run repeatedly:
483
484
485   //============================
486   /////// Temperature /////
487   //============================
488
489   double temp_act = 0.0, press_act = 0.0,hum_act=0.0;
490   signed long int temp_cal;
491   unsigned long int press_cal,hum_cal;
492
493   readData();
494
495   temp_cal = calibration_T(temp_raw);
496   press_cal = calibration_P(pres_raw);
497   hum_cal = calibration_H(hum_raw);
498   temp_act = (double)temp_cal / 100.0;
499   press_act = (double)press_cal / 100.0;
500   hum_act = (double)hum_cal / 1024.0;
501   //Serial.print("TEMP : ");
502   //Serial.print("T");
503   Serial.print(temp_act);
504   Serial.print(",");
505   //Serial.print(" hPa HUM : ");
506   //Serial.print("HUM");
507   Serial.print(hum_act);
508   Serial.print(",");
509   //Serial.print(" DegC PRESS : ");
510   //Serial.print("PRE");
511   Serial.print(press_act);
512   Serial.print(",");
513
514   //Serial.println(" %");
515
516   //delay(1000);
517
518   //============================
519   ///////// Hikari /////////
520   //============================
```

```
521
522   val = analogRead(iPin);
523   //Serial.print("rig");
524   Serial.print(val);
525   Serial.print(",");
526   if(val<920){
527     digitalWrite(oPin, LOW);
528   }else{
529     digitalWrite(oPin, HIGH);
530   }
531   //delay(200);
532
533   //=========================
534   //////// Jinkan ////////
535   //=========================
536
537   // put your main code here, to run repeatedly:
538   if (digitalRead(pirPin) == LOW) {
539     digitalWrite(ledPin, HIGH);
540     //Serial.print("zinkan");
541     Serial.print("1");
542     Serial.print(",");
543
544   } else {
545     digitalWrite(ledPin, LOW);
546     //Serial.print("zinkan");
547     Serial.print("0");
548     Serial.print(",");
549   }
550   //delay(1000);
551
552   //=========================
553   ////////// 9jiku ///////
554   //=========================
555   ////Serial.println
          ("-----------------------------------");
556
557   BMX055_Accl();
558   //Serial.print("Accl= ");
559   //Serial.print("xAccl");
560   Serial.print(xAccl);
561   Serial.print(",");
562   //Serial.print("yAccl");
563   Serial.print(yAccl);
564   Serial.print(",");
565   //Serial.print("zAccl");
566   Serial.print(zAccl);
567   Serial.print(",");
568
569   BMX055_Gyro();
570   //Serial.print("Gyro= ");
571   //Serial.print("xGyro");
572   Serial.print(xGyro);
573   Serial.print(",");
574   //Serial.print("yGyro");
575   Serial.print(yGyro);
576   Serial.print(",");
577   //Serial.print("zGyro");
578   Serial.print(zGyro);
579   Serial.print(",");
580
```

```
581   BMX055_Mag();
582   //Serial.print("Mag= ");
583   //Serial.print("xMag");
584   Serial.print(xMag);
585   Serial.print(",");
586   //Serial.print("yMag");
587   Serial.print(yMag);
588   Serial.print(",");
589   //Serial.print("zMag");
590   Serial.print(zMag);
591   Serial.print(",");
592
593   //delay(1000);
594
595   //============================
596   ////////// Taion ////////
597   //============================
598
599   analog=analogRead(1);
600   V_R0 = analog*3.3/ 1.024;
601   thermo_R=V_0/V_R0* circuit_R0 - circuit_R0;
602   //Serial.println( thermo_R);
603
604   temp=(1000/(1/(0.001*thermo_T0)+log(thermo_R/thermo_R0)*1000/thermo_B)-252);
605   //Serial.print("BT");
606   Serial.print(temp,1);
607   Serial.print(",");
608
609   //============================
610   /////// Shinpaku ////////
611   //============================
612
613   //sendDataToSerial('S', Signal); // goes to sendDataToSerial function
614   //if (QS == true){ // A Heartbeat Was Found
615                            // BPM and IBI have been Determined
616                            // Quantified Self "QS" true when arduino finds a heartbeat
617         // Serial.print("BPM");
618          sendDataToSerial('B',BPM); // send heart rate with a 'B' prefix
619          Serial.print(",");
620          //sendDataToSerial('Q',IBI); // send time between beats with a 'Q' prefix
621          //QS = false; // reset the Quantified Self flag for next time
622   //}
623
624   //delay(20); // take a break
625
626   //============================
627   ////////// GSR //////////
628   //============================
629   long sum=0;
630   for(int i=0;i<10;i++) //Average the 10 measurements to remove the glitch
631       {
632       sensorValue=analogRead(GSR);
633       sum += sensorValue;
634       //delay(5);
635       }
636    gsr_average = sum/10;
637    //Serial.print("GSR");
638    Serial.print(gsr_average);
639  // Serial.print(", ");
640    Serial.println("");
641
```

```
642     delay(5000);
643
644
645
646 }
```

ライフログデータ取得アプリケーションの Arduino2 のソースコード 2 をしめす.

ソースコード A. 2: app2.ino

```
1  //=========================
2  ////////// GPS //////////
3  //=========================
4
5  #include <SoftwareSerial.h>
6
7  // rxPin = 2 txPin = 3
8  SoftwareSerial mySerial(10, 11);
9
10 // NMEAの緯度経度を「度分秒」(DMS)の文字列に変換する
11 String NMEA2DMS(float val) {
12   int d = val / 100;
13   int m = ((val / 100.0) - d) * 100.0;
14   float s = ((((val / 100.0) - d) * 100.0) - m) * 60;
15   return String(d) + "度" + String(m) + "分" + String(s, 1) + "秒";
16 }
17
18 // (未使用)NMEAの緯度経度を「度分」(DM)の文字列に変換する
19 String NMEA2DM(float val) {
20   int d = val / 100;
21   float m = ((val / 100.0) - d) * 100.0;
22   return String(d) + "度" + String(m, 4) + "分";
23 }
24
25 // NMEAの緯度経度を「度」(DD)の文字列に変換する
26 String NMEA2DD(float val) {
27   int d = val / 100;
28   int m = (((val / 100.0) - d) * 100.0) / 60;
29   float s = (((((val / 100.0) - d) * 100.0) - m) * 60) / (60 * 60);
30   return String(d + m + s, 6);
31 }
32
33 // UTC時刻から日本の標準時刻に変換する(GMT+9:00)
34 String UTC2GMT900(String str) {
35   int hh = (str.substring(0,2).toInt()) + 9;
36   if(hh > 24) hh = hh - 24;
37
38   return String(hh,DEC) + ":" + str.substring(2,4) + ":" + str.substring(4,6);
39 }
40
41
42 void setup() {
43   // put your setup code here, to run once:
44
45   /////////// GPS //////////
46
47   mySerial.begin(9600);
48   Serial.begin(115200);
49
50 }
51
```

```
52 void loop() {
53     // put your main code here, to run repeatedly:
54
55   //============================
56   ////////// GPS /////////
57   //============================
58
59    //============================
60   ////////// GPS /////////
61   //============================
62
63   // 1つのセンテンスを読み込む
64   String line = mySerial.readStringUntil('\n');
65
66 // Serial.println(line);
67   if(line != ""){ //# ここは外す
68     int i, index = 0, len = line.length();
69     String str = "";
70
71
72     // StringListの生成(簡易)
73     String list[60];
74     for (i = 0; i < 60; i++) {
75       list[i] = "";
76     }
77
78     //「,」を区切り文字として文字列を配列にする
79     for (i = 0; i < len; i++) {
80       if (line[i] == ',') {
81         list[index++] = str;
82         str = "";
83         continue;
84       }
85       str += line[i];
86     }
87
88     // $GPGGAセンテンスのみ読み込む
89
90     if (list[0] == "$GPGGA") { // # ここは外す
91       // ステータス
92       if(list[6] != "0"){
93         // 現在時刻
94         //Serial.print(UTC2GMT900(list[1]));
95
96         // 緯度
97         //Serial.print(" 緯度:");
98         //Serial.print(NMEA2DMS(list[2].toFloat()));
99         //Serial.print("(");
100        //Serial.print("LON");
101        Serial.print(NMEA2DD(list[2].toFloat()));
102        Serial.print(",");
103
104        // 経度
105        //Serial.print(" 経度:");
106        //Serial.print(NMEA2DMS(list[4].toFloat()));
107        //Serial.print("(");
108        //Serial.print("LAT");
109        Serial.print(NMEA2DD(list[4].toFloat()));
110        Serial.print(",");
111
112        // 海抜
```

```
113          // Serial.print("HEI");
114            Serial.print(list[9]);
115            Serial.print(",");
116            list[10].toLowerCase();
117            //Serial.print("海抜");
118            //Serial.print(list[10]);
119            //Serial.print(", ");
120          }else{
121            //Serial.print("測位できませんでした。");
122          }
123
124          Serial.println("");
125
126            delay(5000);
127       }
128     }
129 }
```

ライフログデータ取得アプリケーションの Raspberrypi のソースコード 3 をしめす.

ソースコード A. 3: app3.py

```
1  import requests
2  import serial
3  import time
4
5  if __name__ == '__main__':
6      gpio_seri = serial.Serial('/dev/ttyACM1', 115200, timeout=10)
7      gpio_seri2 = serial.Serial('/dev/ttyACM0',115200, timeout=10)
8      while 1:
9          gpio_seri.write('get')
10         gpio_seri2.write('get')
11         bio_data = gpio_seri.readline().rstrip()
12         bio_data2 = gpio_seri2.readline().rstrip()
13         bio_data = bio_data.rstrip()+bio_data2
14         bio_data = bio_data.rstrip()
15         print(bio_data)
16         url = "http://172.20.10.6:8080/test1.php"
17
18         response = requests.post(url, data={'bio_csv':bio_data})
```

ライフログデータ取得アプリケーションの php サーバ側のソースコード 4 をしめす.

ソースコード A. 4: app4.php

```
1  <?php
2      $data1 = $_POST['bio_csv'];
3    // $data2 = $_POST['bio2_csv'];
4    // echo $data1;
5    // echo $data2;
6      date_default_timezone_set('Asia/Tokyo');
7      //$data2=(date("Y-m-d H:i:s"),$data1);
8  // $line=implode(',',$data3);
9  // fwite($fp,$line."\n");
10 // fwrite($fp, date("Y-m-d H:i:s")."\t".$data1."\t".$data2."\r\n");
11
12     $file = 'test.csv';
13 // $file2 = 'test2.csv';
14     $current = file_get_contents($file);
15 // $current2 = file_get_contents($file2);
16   // $current .= date("Y-m-d H:i:s");
```

```
17      //$current .= date("Y")+","+date("m")+","+date("d")+","+date("H")+","+date
            ("i")+","+date("s");
18      $current .= date("Y");
19      $current .= ",";
20      $current .= date("m");
21      $current .= ",";
22      $current .= date("d");
23      $current .= ",";
24      $current .= date("H");
25      $current .= ",";
26      $current .= date("i");
27      $current .= ",";
28      $current .= date("s");
29      $current .= ",";
30      $current .= $data1;
31  // $current2 .= $data2;
32
33      file_put_contents($file, $current."\r\n");
34  // file_put_contents($file2, $current2);
35
36  // exec ("node C:\Users\Seiya\Documents\リアルタイムWEB\socket\accept.js"."\t".
        $data1."\t".$data2."\r\n" );
37
38  ?>
```