論文 / 著書情報
Article / Book Information

題目(和文)	ウェアラブルセンサーを用いたリアルタイム作業者動作分析システムとその応用に関する研究
Title(English)	
著者(和文)	北澤正樹
Author(English)	Masaki Kitazawa
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第10705号, 授与年月日:2017年12月31日, 学位の種別:課程博士, 審査員:寺野 隆雄,山村 雅幸,出口 弘,三宅 美博,小野 功
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第10705号, Conferred date:2017/12/31, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

ウェアラブルセンサーを用いた
リアルタイム作業者動作分析システムと
その応用に関する研究

大学院総合理工学研究科

知能システム科学専攻

北澤 正樹

目次

第 1 章　序論

本章では，研究の背景と目的および論文の構成を述べる．

1.1　研究背景

1.1.1　背景と対象とする製造のプロセス

製造業では，消費者からの高品質な製品を安価に購入したいという要求と企業の利益を最大化するという目的を達成するために，企業の運営管理や改善活動を日々行っている[Hopp2000]．その中で，センサーや計算機の高性能化や小型化を利用して工場内のデータを日常的に収集し活用する Industrie4.0 や Industrial internet といった構想が多く提案, 実行されてきている[FMER 2013]，[Evans 2012]，[貝原 2016]，[出口 2016].

工場で収集できるデータには，企業全体の方向性を決める経営戦略や需要を見越した調達や新商品の開発といった中長期データから，生産計画や製造現場の設備制御パラメタといった短期データまで様々な種類がある[ANSI2005]．これらの工場のデータの中でも製品を直接製造している現場のデータが，企業の研究開発活動と生産販売活動の両方に関わり，生産効率の指標として一般的に使われる納期・原価・品質の全てに影響があるため，非常に重要なデータとされている[APSOM2015]（図 1.1）.

1.1.2　製造現場における活動

ここで，製造現場において収集したデータを用いる活動について説明する．製造現場における活動は大きく分けて「管理」と「改善」があり，インダストリアル・エンジニアリング（Industrial engineering：IE）と呼ばれる分野における分類を用いると，製造管理（納期管理), 原価管理, 品質管理, 現場改善活動といったものがあげられる[藤田 1997], [泉 2014].

製造管理には製造スケジューリングや設備保全や作業者育成があり，消費者からの要求納期を安定して満たせるように製造させることを目的として行う活動である．ここで，製

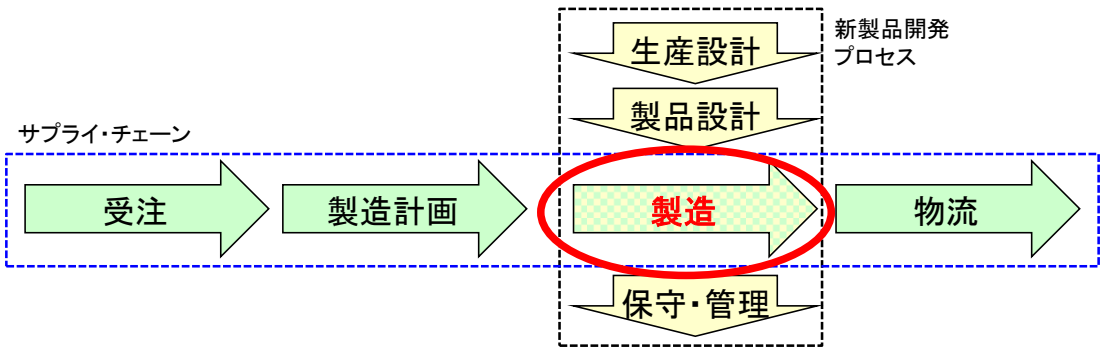


図 1.1　対象とする工場のプロセス

造スケジューリングとは，材料や設備といった資源の制限の範囲内で納期などの目的を達成するには，現在の状況からどのような資源配置を行えばよいのかを明らかにする活動である[久保 2007]. 設備保全とは故障や機能劣化により製造に支障をきたす設備を修理する活動や，設備の使用時間や状態から製造に支障をきたさないように点検や部品交換を行う活動である．作業者育成とは，製造にはじめて携わる作業者が高品質な製品を製造できるように教育したり，製造に慣れてきた作業者がより高度な製造を担当できるようにスキルアップをさせたりするような活動である．また，原価管理には変動費である素材コストの把握や固定費である設備や作業者コストの把握があり，製品の標準原価と実績の差を把握することで製造現場の課題を明確にすることを目的として行う活動である．そして，品質管理には材料のバラつきや製造バラつきの把握があり，消費者に提供される製品の品質を安定させることを目的として行う活動である．

現場改善活動には時間研究と動作研究による製造プロセスの標準化や，稼働分析による作業効率化やムダの削減があり，現在の納期・原価・品質を向上させることを目的として行う活動である．ここで，時間研究と動作研究は[Taylor1911]，[Gilbreath1911]により提唱・実践されてきた手法で，工程において作業者が行う作業をさらに詳細な基本動作に分解した後に，ストップウォッチなどで各動作の時間を測定し，各工程の最適な動作と作業時間を決定する活動である．稼働分析とは作業者の作業時間を，直接材料を加工するような主作業，主作業に付随して発生する運搬のような付随作業，製造の準備のために行わなくてはならない付帯作業，突発的なトラブルにより発生する余裕の 4 つに分解して，ムダの明確化から改善に繋げる活動である．

日本ではこのような製造現場における活動の中でも特に，製造現場の全員がグループごとに一丸となって改善活動を行うボトムアップなアプローチを重視してきた．改善活動では，改善内容を決めるための状況把握と改善実施後の効果評価の 2 点でデータが必要となる．しかしながら，効率化を進めたことによって製造現場に納期や人員の余裕が少なくなり受注している製品の製造だけで現場が手一杯になる，製造に携わる作業者が労働者派遣の利用により入れ替えが激しくなり製造や改善活動の熟練者が少ない，という製造現場の状況から，

1) 製造ラインの課題が材料準備などの製造ライン外の要因であっても状況を把握しきれず，対象とする製造ラインの改善が局所最適に陥って効果がない
2) 従来からの製造ラインを対象とする改善活動の場合，これまでの活動の成果が積み重なっているため，改善によって不良率減少などの目標を達成したように見えても，工数が過剰に必要になるなどの別の評価軸で悪化をまねく
3) 新設の製造ラインを対象とする改善活動の場合，改善活動の効果は得られやすいのにも関わらず，納期遅れや製造トラブルが起きやすいために対応に時間を取られて改善活動や育成の余裕がない

という問題がたびたび発生している．これらは余裕のない製造現場をさらに圧迫する問題

表 1.1　製造現場の収集データと活動の対応

			製造管理			原価管理		品質管理		現場改善活動	
			製造スケジューリング	設備保全	作業者育成	素材コスト把握	製造コスト把握	材料バラつき	製造バラつき	動作研究時間研究	稼働分析
材料	When	使用時期	○	×	×	○	×	×	×	×	×
	Where	使用場所	○	×	×	○	×	×	×	×	×
	What	部品名	○	×	×	○	×	◯	×	×	×
		部品品質	×	×	×	○	×	◯	×	×	×
	How much	使用個数	○	×	×	○	×	◯	×	×	×
設備	When	使用時期	○	○	×	×	◯	×	◯	×	×
	Who	使用者	○	○	×	×	×	×	◯	×	×
	Where	使用設備	○	○	×	×	◯	×	◯	◯	◯
	What	設備パラメタ	×	○	×	×	×	×	◯	×	×
		部品情報	×	○	×	×	◯	×	◯	×	×
		消費電力	○	○	×	×	◯	×	×	×	◯
	How long	稼働時間	○	○	×	×	◯	×	◯	◯	◯
作業者	When	作業時期	○	×	○	×	×	×	◯	×	×
	Who	作業者名	○	×	○	×	×	×	◯	◯	◯
	Where	作業場所	○	×	○	×	×	×	◯	◯	◯
	What	作業内容	○	×	○	×	×	×	◯	◯	◯
	How long	作業時間	○	×	○	×	×	×	◯	◯	◯
		習熟度	○	×	○	×	×	×	◯	×	×

であり，早急な対応が求められる課題である．このような課題を克服する方法の１つとして「製造現場において，製造に関わる全ての要素の状態・動作データを，製品１個単位で，日常的に，もしくは，改善前後などの必要に応じて，容易に収集および分析をできる」という状態を構築することが考えられる．このような状態を構築することで，対象とする製造ラインだけでなく製造現場全体のデータを集めることで局所最適ではなく全体最適に向けた改善活動をできることに加え，作業者に負担をかけることなく製造現場のデータを収集できるため改善活動の実施が容易になるという利点が考えられる．

1.1.3　製造現場のデータ収集

製造現場のデータ収集は，現場の構成要素である材料・設備・作業者それぞれをセンシングすることで行われる．前述の製造現場における活動と計測データの関係は表 1.1 のようにまとめられる．

材料のセンシングでは，いつ，どのような部品を，どこの製造ラインで，いくつ使用されたかを収集することが重要になる．これらのセンシングにはバーコードや Radio

表 1.2　作業者のセンシング方法の比較

			目視		ビデオカメラ 深度カメラ		無線通信 (RFID/Bluetooth/Wi-Fi)		加速度センサー	
作業者	When	作業時期	○	直接記入	○	ログ日時	○	ログ日時	○	ログ日時
	Who	作業者名	○	直接記入	△	解析アルゴリズムの向上次第	○	センサーID	○	センサーID
	Where	作業場所	○	直接記入	○	カメラ位置	○	電波強度による推定	×	推定困難
	What	作業内容	○	直接記入	△	解析アルゴリズムの向上次第	×	該当データなし	○	加速度による推定
	How long	作業時間	○	直接記入	△	解析アルゴリズムの向上次第	△	解析アルゴリズムの向上次第	△	解析アルゴリズムの向上次第
	Cost	計測範囲	×	熟練者1名につき数名まで	△	1台で数m程度	○	1台で 数m～100m程度	×	通信設備次第
		計測コスト	×	常時張り付き	○	自動分析可能	○	自動分析可能	○	自動分析可能

frequency identification（RFID）技術が広く用いられている[三次 2008]，[DSRI2010]，[Altaf2015]．バーコードや RFID タグ貼り付けが必要なため原価の直接増加はあるものの，読み取り作業を行えば定常的に材料の使用量を収集して活用することが可能である．

設備のセンシングでは，いつ，誰が，どの機械を，どのような状態で，どのぐらい使用したかを収集することが重要になる．これらのセンシングには設備にログ保存や信号解析機能を持たせて計測する手法が広く用いられている[Sendon2015]，[鄭 2016]，[竹林 2017]．設備と関係のないプロセスのデータ収集はできないものの，設備やログを保存するサーバーの設定を行えば定常的に設備の稼働時間や稼働パラメタや消費電力を収集して活用することが可能である．

作業者のセンシングでは，いつ，誰が，どこの製造ラインで，どのような状態・動作で，どのぐらい製造進捗かを収集することが重要になる．しかしながら，作業者は自律的に多種の作業を行うためセンシングが難しく，様々な手法が未だ検討されている段階である．作業者のセンシング方法を表 1.2 にまとめる．作業者のセンシングとして従来から製造現場で広く用いられているのは，現場改善活動の熟練者が直接ストップウォッチと共に目視または録画映像を見ながら計測する手法である[Taylor1911]，[Groover2013]．目視による計測では熟練者が直接作業者の状態を記録していけるため作業実績や動作を細かく収集することが可能であるが，熟練者 1 名で多くても一度に数人の作業しか計測できないため，計測コストが膨大で定常的に計測することが難しい．近年ではビデオカメラや深度カメラの映像を自動解析することで計測する手法が現れている[李沢 2012]，[井坂 2016]．ビデオカメラや深度カメラの映像の自動解析では少ない計測コストで作業者の動作を検知することができるが，撮影範囲が数m程度のため工場全体に展開するときに多数のカメラが必要になる，カメラに常に映るように撮影範囲を確保しなければならないのでモノの移動が激しい工場では設置位置の条件検討が必要になる，解析アルゴリズムの向上次第ではあるが現在では撮影範囲内で複数人が同時に作業をしていると個人識別が容易ではないという特徴がある．また，ヒトの状態を推定する手法としては，RFID/Bluetooth/Wi-Fi といった無線

通信技術や加速度センサーを用いる手法もある．RFID/Bluetooth/Wi-Fi といった無線通信では電波受信強度からヒトの屋内位置を推定することが可能で，サービス分野や医療分野などにおける動線分析でよく用いられている[小暮 2008]，[Thiesse2009]，[新村 2012]．しかしながら，電波受信強度からはヒトがどのような動作をしているかは判別できないといった特徴がある．一方で加速度センサーは，身体各所につけた加速度の値からヒトの動作や状態を推定することが可能で，スポーツや高齢者のリハビリなどにおける歩行分析や姿勢分析でよく用いられている[Boyd2011]，[Nagano2014]．しかしながら，ヒトの位置の推測は非常に困難といった特徴がある．

1.2. 研究目的

研究の最終的な狙いは前述の「製造現場において，製造に関わる全ての要素の状態・動作データを，製品 1 個単位で，日常的に，もしくは，改善前後などの必要に応じて，容易に収集および分析をできる」という状態を構築することで，製造現場における作業者の負担を軽減しつつ，全体最適な改善を継続して実施できるようにすることである．本研究ではその狙いの第一歩として，製造現場における作業者のセンシングを対象として，

a) 作業者による作業実績として製品 1 個単位で製造進捗が計測できる
b) 作業者による作業内容として作業履歴と主作業時間が計測できる
c) 複数人が同時に同じ場所にいても作業者ごとに個人特定できる
d) 長時間連続して計測できる

という機能を実現するリアルタイム作業者動作分析システムを提案する．なお，本研究において，製造進捗は製品 1 個を製造するのにかかった所要時間を示し，作業履歴は製品 1 個を製造中の組立と運搬以外の移動を伴う非標準作業（手戻り・工程飛ばし・落下部品の拾得）の有無を示し，主作業時間は製品 1 個を製造するのにかかった所要時間から運搬や移動にかかった時間を除いたものを示すこととする．

本研究では作業者の動作データを収集する手法として，Bluetooth low energy beacon (Beacon)と加速度センサーが組み合わさったウェアラブルセンサーを使用する．ウェアラブルセンサーは日常生活の中で装着しながらセンシングすることを想定したデバイスであるが，

- Beacon と加速度を組み合わせることで作業者の必要な要素を全て計測できる．また，位置推定と動作推定の従来研究が十分にある．
- アンテナとの通信距離が半径 10m 程度と広範囲に展開しやすい
- 電池寿命が数ヶ月～1 年と長い
- 小型で軽量なため，作業者の負担になりにくい
- センサーの低価格化が進んでいる

という特徴があり，現時点ではビデオカメラや深度カメラを用いるよりも容易に本研究の目的を達成できると考えた. 提案システムでは, ウェアラブルセンサーから得られる Beacon

表 1.3　ウェアラブルセンサーによるセンシング

			ウェアラブルセンサー (Bluetooth & 加速度センサー)			無線通信 (RFID/Bluetooth/Wi-Fi)		加速度センサー	
作業者	When	作業時期	○	ログ日時		○	ログ日時	○	ログ日時
	Who	作業者名	○	センサーID		○	センサーID	○	センサーID
	Where	作業場所	○	電波強度による推定		○	電波強度による推定	×	推定困難
	What	作業内容	○	加速度による推定	←	×	該当データなし	○	加速度による推定
	How long	作業時間	△	解析アルゴリズムの向上次第		△	解析アルゴリズムの向上次第	△	解析アルゴリズムの向上次第
	Cost	計測範囲	○	1台で数m～100m程度		○	1台で数m～100m程度	×	通信設備次第
		計測コスト	○	自動分析可能		○	自動分析可能	○	自動分析可能

の電波受信強度データを分析して作業者の位置を推定することで製造進捗を，3 軸加速度データを分析して作業者の移動・停止の動作推定をすることで作業履歴を計測可能にする．

なお，本システムでは作業者の改善活動の負担軽減を狙いとしているが，「管理」でも作業者のデータを用いる活動は多く，作業者のリアルタイムな製造進捗や習熟度を取り入れた動的な製造スケジューリングや進捗予測（例えば[谷水 2003], [八尾 2011], [高橋 2015]），作業者ごとの習熟度を把握したうえでの作業者育成（例えば[舘野 2007], [岩村 2014]）といった研究がなされている．本研究の対象とする範囲ではこのような管理活動への展開は難しいが，今後の機能追加や性能改善で展開できる可能性は十分に考えられる．

本研究は，まず実験により作業者動作の基本データを収集したうえで，計測システムを構築し，基本データから計測のための分析手法やパラメタを決定する．そして，追加実験で作業者動作の評価データを収集し，システムの評価を行うというアプローチで進める．

1.3.　論文構成

本論文の構成は以下の通りである．第 2 章では本研究で対象とする製造形態を定めるとともに，作業者データを収集する実験のための製品やフィールドの設計を示し，合計 3 回行った実験によるデータ収集結果を述べる．第 3 章では提案するシステムの全体像と処理プロセスについて示す．第 4 章では製造進捗の計測のために分析する Beacon の電波受信強度について述べた後に，データ前処理とデータ分析の手法やパラメタと適用した結果を示し，Beacon の電波受信強度に適したデータ前処理と分析手法について明らかにする．第 5 章では作業履歴と主作業時間の計測のために分析する 3 軸加速度について述べた後に，データ前処理とデータ分析の手法やパラメタと適用した結果を示し，3 軸加速度データに適したデータ前処理と分析手法について明らかにする．第 6 章では提案システムの応用例として組立セル製造ラインのエージェント・シミュレーションにシステムによる計測結果を入力することによる有用性を示すとともに，提案システムの本研究範囲での限界や展開性について述べる．最後に第 7 章で結論を述べる．

第 2 章　研究対象と作業者動作データ収集実験

本章では，提案システムの構築および評価に用いる製造データの収集を目的として設計した模擬的な製品およびセル製造ラインの概要と，それを用いた実験の結果について述べる．はじめに工場の製造形態と対象とする製造形態について示した後に，対象とする製品とセル製造ラインの設計を示す．そして，データ収集のために行った実験の環境と，その結果について示す．

2.1.　工場の製造形態と本研究の対象

工場では各製品の必要製造量によって製造形態を変えて対応する．図 2.1 は縦軸に一時間あたりの製造量を，横軸に製品種数とした時に，どのような製造形態をとるか示している[Billo1998]．なお，図 2.1 は目安であり，表示範囲に必ず従うわけではない．

製造量が多い製品はライン生産という，特定製品の製造専用ラインを設置して連続で作り続ける製造形態がとられる．ライン生産はベルトコンベアなどで製品を自動に流し，コンベアの側に並んだ作業者や専用設備が手元にきた製品に対して各自定められた加工や組立を施すことで製造していく方法で，自動車業界や食品業界など様々な場面で使用されている．ライン生産は単一の製品を大量に製造することに適しており，利益も多いため設備投資による自動化で製造効率を向上させやすいという利点があるが，製品の需要が安定していないと供給過多になり在庫を抱えやすい，ライン上流の遅れや品質低下がライン下流まで伝播するため注意が常に必要といった特徴がある．ロット生産は複数の品種を必要ロ

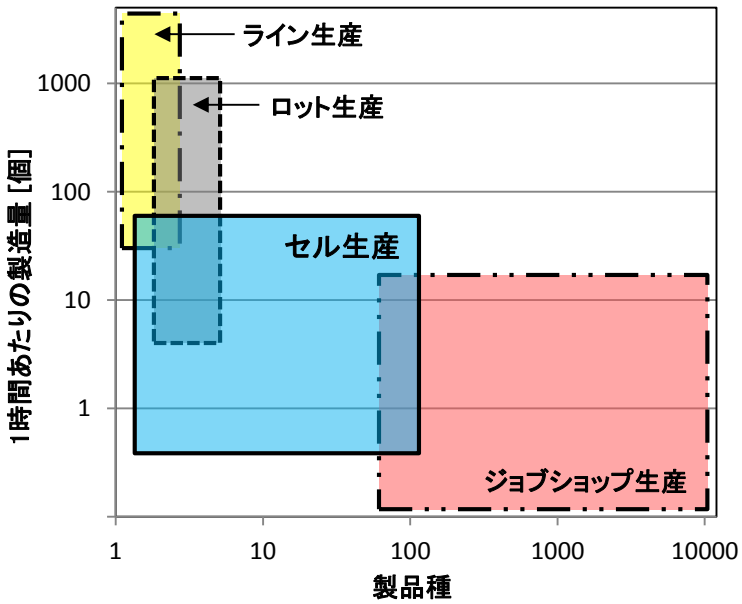
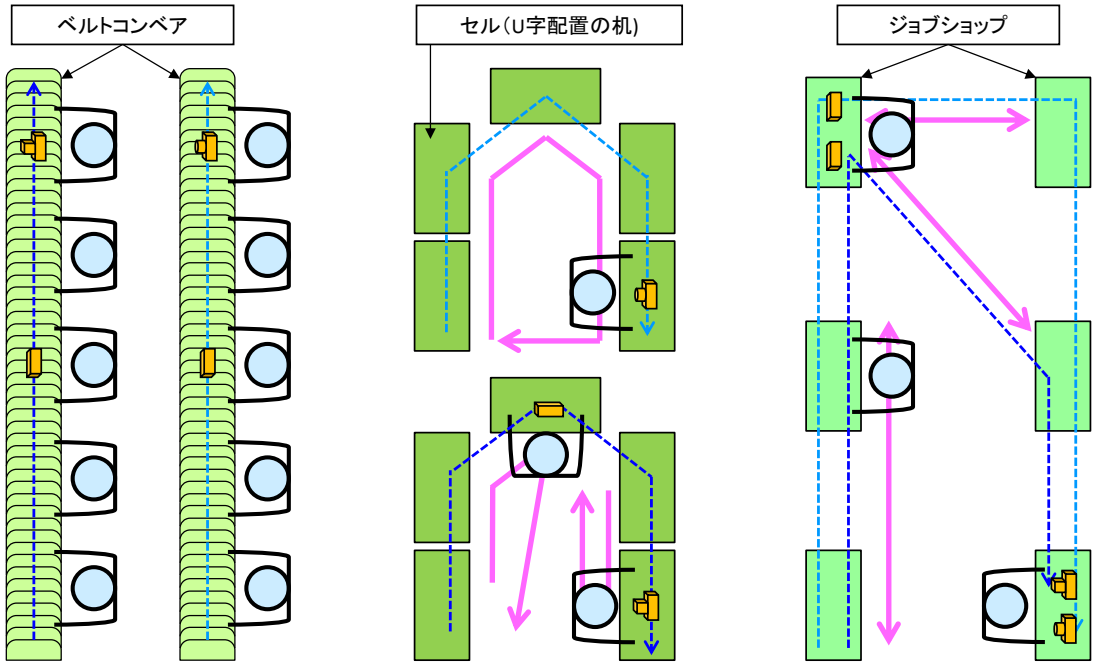


図 2.1　製品種数と製造量による製造形態の違い（[Billo1998]を元に作成）



(a) ライン・ロット生産　(b) セル生産　(c) ジョブショップ生産

図 2.2　工場の各製造形態

ットごとに切り替えながら 1 つの製造ラインで製造する製造形態である．製造中はライン生産と同様の動きをするが，ロットを切り替える時に必要な設備や部品を準備するための段取り替えと呼ばれる準備が必要になる．ロット生産は中品種中量生産に適しており，ライン生産と同様に設備投資による自動化をしやすいという利点があるが，ある品種の製造中は他品種を製造できないため生産計画をうまく調整しなくてはならない，段取り替えを効率よく行わないと製造効率が上昇しないといった特徴がある．セル生産はセルと呼ばれる作業スペース内で 1 人～数人の作業者が複数工程を担当して製品を完成させる製造形態である．セルは作業台を U 字型に並べて形成されることが多く，作業者はセル内部を歩いて製品を自分で運搬しながら加工や組立を施すことで製造していく方法で，情報機器業界や家電業界などの多くの場面で使用されている．各セルで製造する製品を変更することが可能なセル生産は多品種少量生産に適しており，セル数の増減で需要の変動にも対応しやすいという利点があるが，各作業者が複数工程を担当するためのスキルを習得しなければならない，担当する作業者のスキルレベルに依存するため製造効率の安定化が難しい，設備投資による自動化をしにくいといった特徴がある．ジョブショップ生産は加工や組立を機能ごとに集めたジョブショップを形成し，多種の製品を製造手順に応じて必要なジョブショップに運搬し，製造していく製造形態である．ジョブショップは複雑な加工ができる設備であることが多く，作業者は特定のジョブショップを担当して製品を製造していく．ジョブショップの組み合わせ方により製造できる製品が変わるため顧客の個別注文対応に適しており，単一作業の管理はしやすくなるという利点はあるが，工場全体では製品ごと

に製造手順が変わるため現状把握が難しい，多種の製品に対応するため段取り替えが多く発生し製造効率の向上が難しいといった特徴がある．

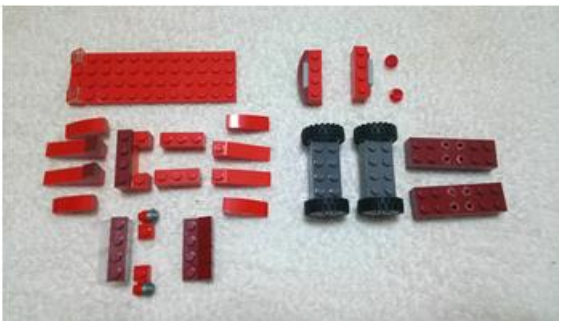
本研究ではこれらの製造形態の中からセル生産形態の，特に部品を組み立てて製品を製造する組立セル製造ラインに焦点をあてて研究を行う．理由としては 1) セル生産は多品種少量生産に適しており近年の消費者の多様化に対応しやすく有用性が増している，2) 計測対象である作業者による作業が主である，3) 設備導入が難しいため作業者の動きから製造進捗や作業履歴の計測することが重要である，4) 作業範囲が狭いため従来の位置推定精度での製造進捗が困難という技術課題がある，という 4 点より本研究の差異が明確になると考えたためである．

2.2. 実験製品の設計

組立セル製造ラインの実験で製造する製品は，レゴブロックの販売キットの中から抜粋して準備した[LEGO2016]．レゴブロックを用いて工場のセル製造ラインの実験を行う手法は，製造業では研究やビジネス検収など広く使われている[Johansen1995]，[Dong2014]，[Tanimizu2015].

準備をした製品はクルマの組立（1 人用・2 人協力用），クルマの分解（1 人用・2 人協力用），ペンギンの組立(1 人用)，ハウスの組立(1 人用)の 4 種類 6 通りである．各製品の完成形を図 2.3 に，作業手順を表 2.1 に示す．各製品の部品ブロックは，どの製品の作業数も同

(a) クルマの組立製品完成形



(b) クルマの分解製品完成形

(c) ペンギンの組立製品完成形

(d) ハウスの組立製品完成形

図 2.3　レゴブロックを用いた実験用製品

表 2.1　各製品の作業手順と標準作業時間

(a) 1 人用製品 4 種類

作業番号	基本作業名	クルマ組立		クルマ分解		ペンギン組立		ハウス組立	
		部品詳細	標準作業時間 [s]	部品詳細	標準作業時間 [s]	部品詳細	標準作業時間 [s]	部品詳細	標準作業時間 [s]
1	取り出し	ベース	1.14	製品	1.14	ベース	1.14	ベース	1.14
2	部品I作業	フロントボディ	9.84	テールランプ バンパー	8.88	ボトムボディ	7.38	煙突下部	2.46
3	運搬		1.20		1.20		1.20		1.20
4	部品II作業	ミドルボディ	7.38	タイヤ	4.32	リアボディ	7.38	左壁	14.76
5	運搬		1.20		1.20		1.20		1.20
6	部品III作業	リアボディ	9.84	ボトムボディ	5.64	フロントボディ	2.46	右壁	9.84
7	運搬		1.20		1.20		1.20		1.20
8	部品IV作業	サイドミラー	4.92	フロントガラス リアウィング	4.56	サイドボディ	4.92	ドア 窓	4.92
9	運搬		1.20		1.20	※ペンギンは部品Vなし	2.28		1.20
10	部品V作業	フロントガラス リアウィング	4.92	サイドミラー	4.32			上壁	4.92
11	運搬		1.20		1.20				1.20
12	部品VI作業	バンパー	5.16	リアボディ	9.00	アーム	4.92	屋根下部	12.30
13	運搬		1.20		1.20		1.20		1.20
14	部品VII作業	タイヤ	4.92	ミドルボディ	6.66	フット	5.16	屋根上部	7.38
15	運搬		1.20		1.20		1.20		1.20
16	部品VIII作業	ボトムボディ	5.16	フロントボディ	9.00	ヘッド	2.70	煙突上部	9.84
17	完成品納品	完成品	0.78	ベース	0.90	完成品	0.78	完成品	0.78
18	移動		1.68		1.68		1.68		1.68
合計標準作業時間（標準サイクルタイム）			**64.14**		**64.50**		**46.80**		**78.42**

(b) 2 人協力用製品 2 種類

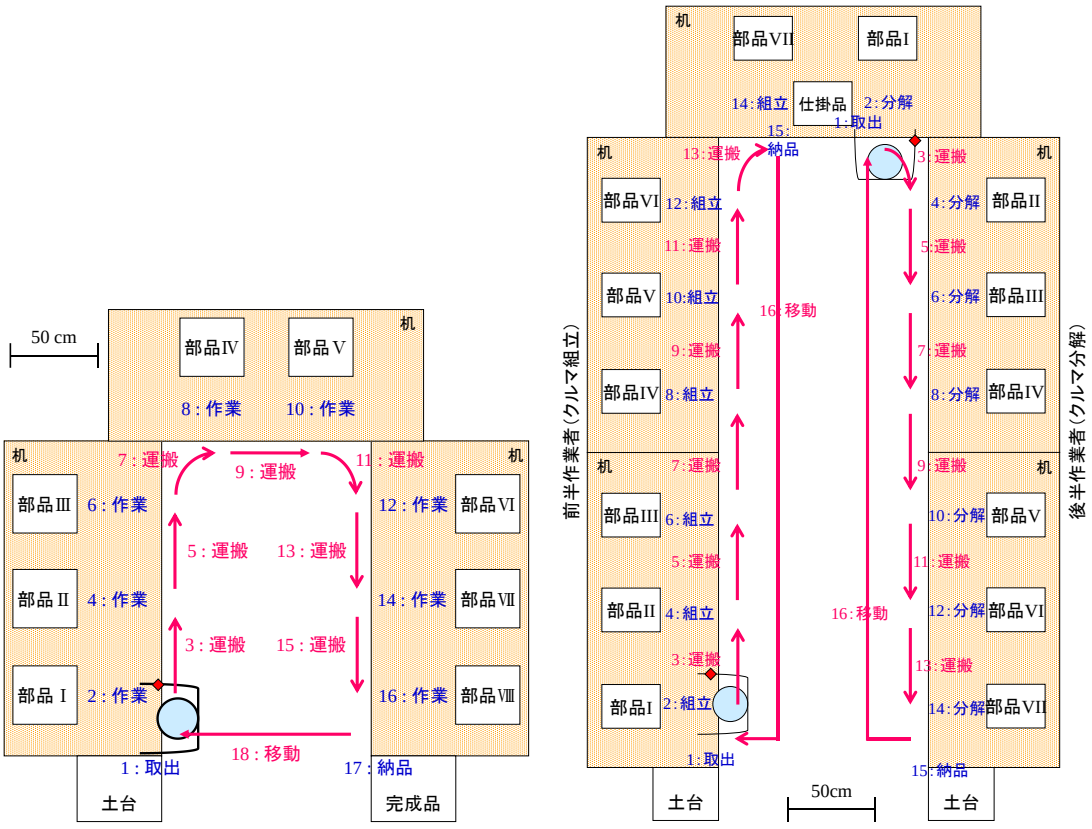
作業番号	基本作業名	クルマ組立		クルマ分解	
		部品詳細	標準作業時間 [s]	部品詳細	標準作業時間 [s]
1	取り出し	ベース	1.14	完成品	1.14
2	部品I作業	フロントボディ	9.84	テールランプ バンパー	8.88
3	運搬		1.20		1.20
4	部品II作業	ミドルボディ	7.38	タイヤ ボトムボディ	9.96
5	運搬		1.20		1.20
6	部品III作業	リアボディ	9.84	フロントガラス リアウィング	4.56
7	運搬		1.20		1.20
8	部品IV作業	サイドミラー	4.92	サイドミラー	4.32
9	運搬		1.20		1.20
10	部品V作業	フロントガラス リアウィング	4.92	リアボディ	9.00
11	運搬		1.20		1.20
12	部品VI作業	バンパー	5.16	ミドルボディ	6.66
13	運搬		1.20		1.20
14	部品VII作業	タイヤ ボトムボディ	10.08	フロントボディ	9.00
15	完成品納品	完成品	0.78	ベース	0.90
16	移動		3.12		3.12
合計標準作業時間（標準サイクルタイム）			**64.38**		**64.74**

程度になるように切削や接着を施した．なお，組立セル製造ラインを対象としていてクルマに組立と分解の 2 種類を準備したのは，部品数の制約がある中で連続して長時間の製造実験を可能にするためである．

さらに，各製品にはインダストリアル・エンジニアリングの Ready work factor 法(RWF)を利用して，製造の標準作業時間を設定した．RWF はある作業の所要時間をどのぐらいにすれば良いかを定める手法の 1 つであり，製造するときの作業を動作単位に分解した後に，あらかじめ定められた時間値表から寸法や重量にあわせて各動作の所要時間を求めて集計することで標準作業時間を導出することができる[Tanner1990]，[吉田 2016]．なお，標準作業時間を合計して製品 1 個あたりにしたものを標準サイクルタイムと呼ぶ．標準サイクルタイムは生産計画時の設定値にしたり，作業者のスキルレベルを確認したりと有用な基準である．本研究で準備した製品の標準作業時間と標準サイクルタイムは表 2.1 に示した．

2.3. 実験フィールドの設計

作業者が歩き回りながら製品を製造するフィールド：セルは，実際の工場のサイズを模した寸法で U 字型に机と部品を配置して設計した．設計したフィールドを図 2.4 に示す．図内に記載された数字と作業名は，表 2.1 の各作業番号と対応している．



(a) 1 人用セル　　(b) 2 人協力用セル

図 2.4　フィールド設計

2 人協力用セルでは，クルマの組立を担当する前半作業者が完成品を仕掛品置場に納品すると，後半作業者が完成品を取り出して分解を行うようになっている．そのため，後半作業者は前半作業者が製品完成させるまで待機する必要があるが，標準サイクルタイムは 64.38 秒と 64.74 秒でわずかに後半作業者の方が遅いため，設計上は待ち時間が発生しないようになっている．

2.4. 使用ウェアラブルセンサー

本研究では作業者の動作データを，ウェアラブルセンサーを用いて収集する．製造進捗と作業履歴を計測するためには，センサーから収集したデータでそれぞれの作業者が，いつ，フィールド内のどこにいて，作業手順どおりの動きをしているか分析できなくてはならない．そのために，本研究におけるウェアラブルセンサーは以下の機能を持つものとする．

- センサーの通信時刻を記録できる
- 各センサーに ID を設定して，記録できる
- Bluetooth low energy beacon(Beacon)の電波受信強度(Received signal strength indicator : RSSI)の記録ができる
- 3 軸加速度センサーの値の記録ができる
- 高頻度でデータの記録ができる
- サーバーに格納した生データを利用できる

ここで，センサーの通信時刻は「いつか」を，各センサーの ID は「どの作業者か」を，Beacon の RSSI は「フィールド内のどこか」を，3 軸加速度センサーの値は「作業手順どおりの動きをしているか」を把握するためである．

本研究ではこれらの機能を満たすセンサーの中から，ベイシスイノベーション社の StickNFind を利用した（図 2.5）．使用した StickNFind は Beacon と 3 軸加速度以外に温度計も備えながら，センサー約 5000 円/個，アンテナ約 25000 円/個とリーズナブルな値段で購入可能であったため利用した．さらに，センサーからサーバーまでの接続形態を図 2.6 に示

図 2.5　ウェアラブルセンサーとアンテナ

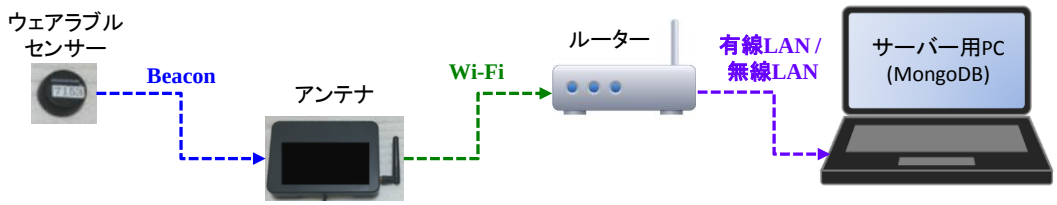


図 2.6　ウェアラブルセンサーからサーバーの接続形態

す．センサーは設定した頻度で Beacon のパケットを発信しつづけており，アンテナの組込アプリが受信したパケットを元に Wi-Fi 経由でローカルサーバーにデータを蓄積していく．なお，遮蔽物がない状態でのセンサーとアンテナ間の通信可能距離は約 10 m である．

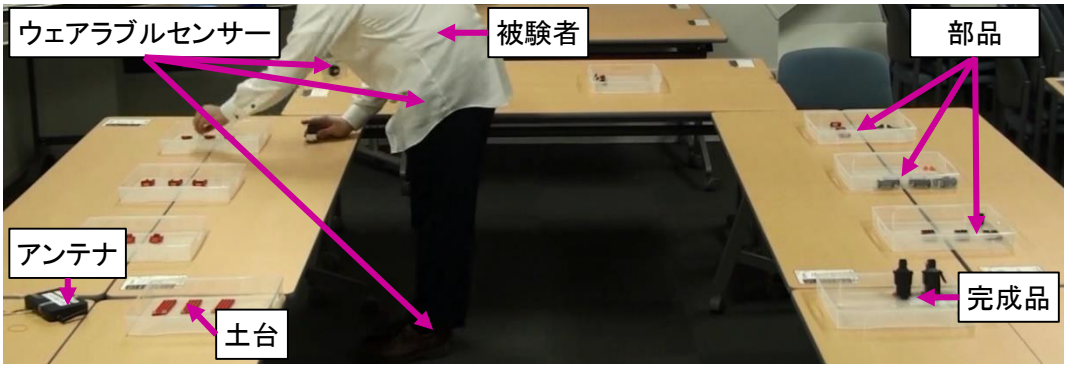
2.5.　実験環境

設計した製品とフィールド，ウェアラブルセンサーを用いて合計 3 回の実験を行い，作業者動作データを収集した．各実験の設定を表 2.2 に，実験場面を図 2.7 に示す．

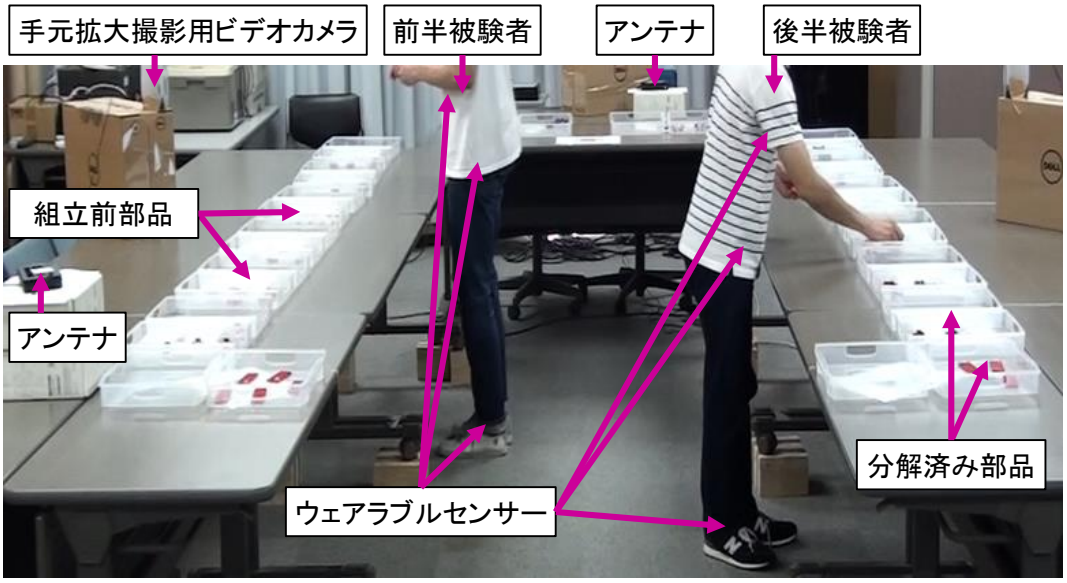
第 1 回実験は 20 代の男性 3 名女性 1 名と 30 代の男性 1 名の被験者 5 人で実施し，被験者それぞれ 3 種類の製品を 10 個ずつ合計 30 個製造した．各被験者は製品の作り方や製造

表 2.2　作業動作データ収集実験環境

実験回	1	2	3
日時	2016年3月7日	2016年9月5日	2016年9月6日
場所	東京理科大学講義室	東京工業大学会議室	
フィールド	1人用セル		2人協力用セル
製品	3種類 ・クルマ組立(1人用) ・ペンギン組立(1人用) ・ハウス組立(1人用)	2種類 ・クルマ組立(1人用) ・クルマ分解(1人用)	2種類 ・クルマ組立(2人用) ・クルマ分解(2人用)
製造 ノルマ	各10個　計30個	60分	
部品 準備/供給	内段取 種類変更時は待機	外段取　常に部品あり	
被験者 人数	5名	6名 組立3名 / 分解3名	3ペア6名
センサー 装着箇所	3箇所　左胸前 / 腰右側 / 右足首		
アンテナ 設置箇所	1箇所　土台置場		2箇所 土台置場 / 部品I置場
収集データ	・作業者動作の センサーデータ ・セル俯瞰動画	・作業者動作のセンサーデータ ・セル俯瞰動画 ・手元拡大動画 ・実験前後の疲労度アンケート4種類 [Chalder1993][酒井2002] [産業疲労研究会2016][Scott1976]	



(a) 第 1 回実験時の 1 人用セル



(b) 第 3 回実験時の 2 人協力用セル

図 2.7　実験環境と実験の様子

時の動き方を事前に学習した状態で取り組んでもらった．被験者は実験時にセンサーを左胸前，腰右側，右足首に装着して作業者動作データを収集できるようにした．ウェアラブルセンサーは手首に装着することが多いが，セル製造ラインの作業者は接触による部品の破損や作業速度の低下を防ぐ目的で手首への着用を嫌うことがあるため，除外した．各被験者には製造の開始前に 30 秒程度，製造開始位置に停止させ，キャリブレーション用のデータを収集した．製品の種類切替は被験者にはその場で待機させ，その間に被験者以外で部品を準備するという内段取手順で行った．また，センサーデータ以外にビデオカメラで実験を撮影したセル俯瞰動画も収集した．

第 2 回実験は 20 代男性がクルマ組立 3 人とクルマ分解 3 人で合計 6 人，第 3 回実験は第 2 回実験と同じ被験者がクルマ組立担当とクルマ分解担当でペアとなった 3 組合計 6 人で実施し，被験者はそれぞれの担当製品を約 60 分製造し続けた．第 2 回実験と第 3 回実験の 6 名は同一人物で，被験者の対応は表 2.3 に示すとおりである．各被験者は第 1 回と同様に製

表 2.3　第 2 回実験と第 3 回実験の同一被験者の対応表

担当	第2回実験	第3回実験
クルマ（組立）	被験者2-1	被験者3-1-1
	被験者2-2	被験者3-2-1
	被験者2-3	被験者3-3-1
クルマ（分解）	被験者2-4	被験者3-2-2
	被験者2-5	被験者3-3-2
	被験者2-6	被験者3-1-1

品の作り方や製造時の動き方を事前に学習し，実験時はセンサーを左胸前，腰右側，右足首に装着して作業者動作データを収集できるようにした．また，第 1 回と同様に製造開始前にキャリブレーション用のデータも収集した．製品の部品は各 10 個分であったが，第 2 回実験時は被験者が製品を完成させたらすぐに回収してフィールド外で組立もしくは分解することで，フィールド内には常に部品がある状態を維持して製造が止まらないようにした．第 3 回実験時は後半被験者が分解した部品を前半被験者に供給することで，フィールド内には常に部品がある状態を維持して製造が止まらないようにした．ただし，前半作業者の作業が遅れて完成品供給がない場合は，後半作業者は手順 1 の作業位置で待機をさせた．また，センサーデータとセル俯瞰動画に加えて被験者の手元の作業様子を撮影した手元拡大動画と，実験の前後に被験者の疲労度を測定するために Chalder fatigue scale [Chalder1993]，自覚症しらべ[酒井 2002]，疲労部位しらべ[産業疲労研究会 2016]，Visual analogue scale [Scott1976]の 4 種類のアンケートを実施した．本研究は疲労に関する分析は対象外であるが，これらの疲労度アンケートと作業者動作データの分析を結びつけることで，今後，動作データからの作業者の疲労度推定に繋げることができると考えられる．

2.6.　実験結果

収集した作業者動作データの一部を表 2.4 に示す．データ 1 行がウェアラブルセンサー1 個の計測した数値に対応しており，必要機能の通信時刻，センサーID，Beacon の RSSI，3

表 2.4　収集した作業者動作データ

received_at	mac	rssi	uuid	major	minor	temp.	Voltage	x_accel	y_accel	z_accel
2016-03-07T01:42:57.702Z	C0:0C:67:1D:CF:92	-86	3403	318	7153	24.25	2780	98	-228	-19
2016-03-07T01:42:57.754Z	D0:BD:8F:AC:5B:A7	-70	8403	318	7141	26.25	2750	185	44	171
2016-03-07T01:42:57.804Z	C0:0C:67:1D:CF:92	-78	3403	318	7153	24.25	2780	98	-228	-19
2016-03-07T01:42:57.858Z	CF:40:35:19:3F:C6	-56	8403	318	7152	26.75	2890	-239	47	69
2016-03-07T01:42:57.909Z	CF:40:35:19:3F:C6	-89	8403	318	7152	26.75	2890	-239	47	69
2016-03-07T01:42:58.113Z	D0:BD:8F:AC:5B:A7	-57	8403	318	7141	26.25	2750	188	42	166
2016-03-07T01:42:58.467Z	CF:40:35:19:3F:C6	-63	8403	318	7152	26.75	2890	-240	49	67
2016-03-07T01:42:58.538Z	C0:0C:67:1D:CF:92	-93	3403	318	7153	24.25	2780	98	-228	-19
2016-03-07T01:42:58.608Z	C0:0C:67:1D:CF:92	-92	3403	318	7153	24.25	2780	98	-228	-19
2016-03-07T01:42:58.695Z	C0:0C:67:1D:CF:92	-90	3403	318	7153	24.25	2780	98	-228	-19
2016-03-07T01:42:58.754Z	D0:BD:8F:AC:5B:A7	-57	8403	318	7141	26.25	2750	186	43	167
2016-03-07T01:42:58.809Z	C0:0C:67:1D:CF:92	-73	3403	318	7153	24.25	2780	98	-228	-19
			・							
			・							
			・							

通信時刻
センサーのハード側ID
BeaconのRSSI
センサーのソフト側ID (3段階)
温度
電池残量
3軸加速度

軸加速度に加えて，サーバー側のインデックスやセンサーの MAC アドレス，気温や電池残量が記録されている．なお，3 軸加速度は 256 を 1.0 g (≒9.8 m/s^2)としたときの値が記載されている．

各回のセル俯瞰動画を目視で確認し，各作業者の製造進捗としてサイクルタイムの実績を，作業履歴として製造時の異常動作の有無を，作業内容の詳細として主作業である組立時間を計測した．このとき，多くの作業者においてベース取り出し時と完成品納品時に，図 2.8 に示すような標準作業にない移動（運搬）を追加していたことが確認された．しかしながら，本研究における作業履歴は「製品 1 個を製造中の組立と運搬以外の移動を伴う非標準作業（手戻り・工程飛ばし・落下部品の拾得）の有無」としているため，これらの追加移動は異常動作には含まないこととした．計測した結果を表 2.5～表 2.8 に示す．本研究では以後，この分析結果を製造進捗と作業履歴と作業内容の真値として扱う．

表 2.5 で第 1 回実験の製品種類切替時の待機時間と，表 2.6 で第 3 回実験の前半被験者が製品 1 個目を完成させるまでの後半被験者の待機時間は除外した結果を示している．作業履歴について第 3 回実験で部品落下などの異常動作が発生しなかったのは，被験者が第 2 回実験をとおして習熟したことによる影響と考えられる．また，表 2.5～表 2.7 に記載の平均動画分析所要時間は製造進捗と作業履歴の目視確認に要した時間であり，作業内容の詳細な時刻の計測には製品 1 個あたり約 4～5 分の時間がかかり，目視確認の計測コストの膨大さを示している．また本研究では製品と標準作業の設計上，作業者の停止時間と主作業の時間が等しくなり，残りの作業者の移動時間と付随作業の時間が等しくなる．

なお，本実験ではいくつか予期せぬ実験条件の変更が生じた．第 2 回実験の被験者 2-4 の合計作業時間がノルマの 60 分よりも大幅に短い 50 分になっているのは，外段取作業にミスがあり途中で部品が供給できず連続製造が不可能となったためである．また，データ収集頻度が第 1 回では 3 個のセンサーで 1 秒あたり 14 回となっていたが第 2 回と第 3 回では 1 秒あたり 5.1 回と減少しているのは，第 2 回実験の準備中にアンテナのアプリケーションのアップデートにより設定がリセットされてしまったためである．

(a) 取り出し動作時の追加移動　　(b) 納品動作時の追加移動

図 2.8　実験で確認された標準作業以外の追加移動

表 2.5　第 1 回実験の各被験者実験結果

製品種類	製品数	被験者1-1		被験者1-2		被験者1-3		被験者1-4		被験者1-5	
		時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無
クルマ組立	1	102.41		58.36		51.99		69.84		70.47	
	2	125.85	異常	89.65		48.25		63.73		65.57	
	3	102.37		61.43		44.44		61.26		125.93	異常
	4	121.76		60.39		46.28		65.40		72.30	
	5	77.28		59.53		49.92		67.46		80.82	
	6	86.88		57.16		48.78		72.21		74.40	
	7	88.36		49.95		46.88		69.90	異常	71.24	
	8	92.86	異常	53.15		47.65		59.69		66.13	
	9	91.65		46.85		62.09		64.60		69.67	
	10	95.81		47.28		44.05		53.49		58.59	
ペンギン組立	11	101.03		62.59		35.04		57.26	異常	57.02	異常
	12	59.13		39.27		30.13		71.84		46.08	
	13	60.83		48.85		32.43		51.62		49.58	
	14	60.59		37.31		30.90		59.39		40.75	
	15	56.79		31.56		28.32		36.94		43.24	
	16	79.61		35.27		27.60		40.87		38.80	
	17	51.12		41.77		32.90		51.82		45.68	
	18	57.89		32.00		32.73		41.98		41.01	
	19	54.19		32.20		30.90		72.47		38.14	
	20	45.88		34.20		27.43		47.68		40.37	
ハウス組立	21	138.68		69.34		76.01		92.03		122.23	
	22	101.40		67.90		62.39		106.37	異常	99.00	
	23	106.47		54.68		48.27		101.27		96.22	
	24	92.66		64.74		54.05		93.06	異常	101.21	
	25	90.36		64.06		54.93		84.68		85.68	
	26	91.59		58.36		54.65		61.63		73.81	
	27	86.62		61.66		49.12		90.26	異常	72.54	
	28	88.09		55.86		50.51		91.22	異常	85.15	異常
	29	98.76		61.29		48.52		93.69	異常	80.15	
	30	88.86		58.03		52.12		100.30		85.08	異常
合計異常回数		2		0		0		7		4	
合計作業時間 [s]		2595.78		1594.69		1349.28		2093.96		2096.86	
平均サイクルタイム [s]		86.53		53.16		44.98		69.80		69.90	
平均サイクル（クルマ）[s]		98.52		58.38		49.03		64.76		75.51	
平均サイクル（ペンギン）[s]		62.71		39.50		30.84		53.19		44.07	
平均サイクル（ハウス）[s]		98.35		61.59		55.06		91.45		90.11	
データ数		34035		21978		18667		31267		30403	
平均データ収集頻度		14 データ/s（センサー3個使用のため1個あたりは4.67 データ/s）									
平均動画分析所要時間		25 s/個									

表 2.6　第 2 回実験の各被験者実験結果（その 1）

製品数	被験者2-1		被験者2-2		被験者2-3		被験者2-4		被験者2-5		被験者2-6	
	クルマ(組立)						クルマ(分解)					
	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無
1	73.64		74.74		54.76		48.62		57.82		63.87	
2	73.27		59.66		59.32		39.20		55.16		66.46	
3	72.28		59.33		59.26		49.92		53.82		55.49	
4	69.80		55.19		55.62		36.17		47.38		58.49	
5	78.68		57.39		58.53		30.53		53.09		56.26	
6	78.61		57.75		63.26		37.30		55.32		55.69	
7	78.31		55.83		72.54		34.84		54.95		57.96	
8	87.16		63.09		61.66		36.37		55.26		49.95	
9	71.37		55.66		61.27		35.67		51.58		56.25	
10	75.14		59.29		60.52		33.06		52.96		55.29	
11	71.74		56.22		59.93	異常	35.47		50.88		51.72	
12	67.13		57.63		55.79		40.64		48.25		53.82	
13	69.84		57.72		56.15		45.38		44.38		51.19	
14	71.63		54.86		63.83		36.24	異常	39.07		52.28	
15	73.58		56.55		55.43		35.01	異常	50.38		51.95	
16	72.87		59.60		55.42		44.41		48.98		46.65	
17	70.14		55.42		56.35		43.91		49.02		49.12	
18	83.92		59.79		53.83		40.51		46.01		40.04	
19	76.30		53.62		55.15		45.58		45.61		45.01	
20	70.68		57.46		54.99		44.74		48.39		43.11	
21	65.06		55.52		54.49		40.78		48.24		43.11	
22	70.20		54.59		58.69		42.31		55.46		39.97	
23	66.97		51.92		60.13		45.71		48.21		45.08	
24	68.40		55.25		54.55		47.11		47.99		45.01	
25	68.54		56.69		59.26		42.64		48.78		41.01	
26	85.95		58.90		52.15		44.55		52.52		41.07	
27	72.57		57.99		51.19		45.38		52.85		43.01	
28	68.80		59.86		52.28		46.68		52.55		44.28	
29	69.67		53.92		53.59		44.37	異常	54.22		42.84	
30	72.01		63.19		53.58		40.44		45.52		42.38	
31	70.87		56.43		49.89		45.35		51.58		50.25	
32	68.87		53.18		60.83		40.94		43.91		45.11	
33	63.50		54.62		52.98		43.34		44.65		44.64	
34	71.53		52.32		66.10	異常	51.05		50.61		42.01	
35	77.21		50.15		55.36		42.95		39.38		41.94	
36	70.41		52.66		52.72		44.04		35.86		45.52	
37	64.60		52.85		56.25		42.94		45.12		37.17	
38	68.83		47.01		53.96		43.51		37.07		44.64	
39	58.19		51.82		55.92		51.89		45.21		43.38	
40	67.34		55.12		56.52		47.65		44.08		41.91	
41	67.20		52.12		53.96		51.81		45.84		43.34	
42	66.63		53.55		50.18		48.49		39.51		53.85	
43	64.97		51.89		56.89		44.54		44.11		41.88	
44	59.99		52.79		51.65		50.88		46.61		46.01	
45	65.00		55.78		55.06		54.89	異常	46.45		41.48	
46	61.59		54.63		51.65		47.82		52.48		38.74	
47	59.63		50.61		55.22		45.18		53.86		44.01	
48	64.70		52.02		51.98		49.85		48.51		42.17	
49	60.69		50.59		52.59		50.48		43.65		39.87	
50	66.20		50.45		51.89		48.91		47.04		41.31	
51	58.79		46.74		52.01		47.82		44.98		35.84	
52	55.49		47.05		53.02		51.18		42.04		39.14	
53	62.06		50.15		50.32		49.05		43.38		39.54	
54	64.17		47.05		53.29		45.15		42.94		37.70	
55	67.93	異常	51.85		48.88		48.48		43.35		41.34	
56	61.06		45.94		56.76		44.14		41.94		38.91	
57			54.59		53.05		51.62		39.04		42.97	
58			50.95		53.29		49.35		38.84		36.31	
59			49.09		54.08		47.41		36.03		35.23	
60			45.91		54.29		47.42	異常	34.44		37.14	

表 2.6　第 2 回実験の各被験者実験結果（その 2）

製品数	被験者2-1		被験者2-2		被験者2-3		被験者2-4		被験者2-5		被験者2-6	
	クルマ(組立)						クルマ(分解)					
	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無
61			47.85		51.45		52.89		35.63		38.47	
62			49.81		53.72		47.51		36.10		35.80	
63			50.42		53.02		45.88		37.84		34.71	
64			50.32		50.42		45.68		38.37		33.83	
65			54.68		53.95		45.64		40.56		36.37	
66			49.59		52.96		45.65		36.55		36.27	
67			50.21		57.62		42.67		37.71		35.80	
68							38.88		38.80		35.74	
69									34.91		37.30	
70									37.00		38.64	
71									35.37		38.87	
72									31.03		36.44	
73									35.90		33.33	
74									38.87		36.80	
75									38.14		34.30	
76									35.44		35.37	
77									35.13		33.70	
78									38.74		39.61	
79									33.90		32.43	
80									41.14		32.03	
81											31.30	
82											33.37	
83												
84												
85												

製造個数	56	67	67	68	80	82
合計異常回数	1	0	2	5	0	0
合計作業時間 [s]	3881.71	3627.42	3717.25	3026.47	3564.29	3522.19
平均サイクルタイム [s]	69.32	54.14	55.48	44.51	44.55	42.95

データ数	19143	17686	18373	18399	17611	17220
平均データ収集頻度	5.1 データ/s（センサー3個使用のため1個あたりは1.7 データ/s）					

平均動画分析所要時間	32 s/個

表 2.7　第 3 回実験の各被験者実験結果（その 1）

製品数	ペア3-1				ペア3-2				ペア3-3			
	被験者3-1-1		被験者3-1-2		被験者3-2-1		被験者3-2-2		被験者3-3-1		被験者3-3-2	
	クルマ(組立)		クルマ(分解)		クルマ(組立)		クルマ(分解)		クルマ(組立)		クルマ(分解)	
	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無
1	55.19		55.05		49.92		48.48		45.18		50.72	
2	55.15		57.76		46.91		48.18		50.55		47.62	
3	57.56		60.76		49.25		46.55		47.51		48.04	
4	60.80		56.86		48.18		52.45		47.95		50.75	
5	58.32		55.82		48.95		51.65		51.42		51.72	
6	54.86		53.62		52.39		45.02		51.58		53.29	
7	53.62		57.76		47.38		47.74		53.12		52.32	
8	57.69		55.59		47.71		48.12		52.65		53.75	
9	53.42		53.62		47.98		50.35		53.89		50.69	
10	56.32		57.29		50.35		45.81		50.38		52.85	
11	56.29		59.66		45.85		48.98		52.69		54.72	
12	59.69		55.58		48.95		56.29		55.02		55.72	
13	56.16		59.26		56.25		49.49		55.76		48.42	
14	59.12		59.80		49.65		48.78		48.01		51.48	
15	59.96		58.15		49.22		52.12		51.85		49.82	
16	57.96		55.23		48.11		51.01		49.82		50.18	
17	55.26		55.35		49.15		45.88		50.18		50.62	
18	55.39		55.36		46.28		56.16		50.56		51.12	
19	55.72		72.04		48.82		47.45		50.85		50.38	
20	49.61		29.96		52.18		49.21		50.81		51.15	
21	52.09		61.26		45.92		37.04		51.22		50.89	
22	60.56		51.15		43.37		43.48		50.38		49.75	
23	52.32		54.36		47.92		45.94		50.09		49.91	
24	54.05		50.95		46.04		46.22		49.88		49.62	
25	47.55		49.45		45.85		46.24		49.72		49.85	
26	52.05		49.11		46.34		46.18		49.81		49.38	
27	50.12		55.12		46.52		49.32		49.52		51.18	
28	54.82		52.56		49.51		46.24		51.85		49.75	
29	52.55		57.09		45.92		50.39		49.02		54.16	
30	57.36		57.55		46.54		43.57		54.22		47.08	
31	57.12		57.53		44.28		40.98		46.88		49.42	
32	57.66		49.78		43.98		46.88		49.58		52.61	
33	47.35		54.46		47.38		44.68		52.29		47.95	
34	57.09		51.08		43.84		47.58		47.98		53.12	
35	47.05		49.95		48.15		47.91		53.18		47.25	
36	48.08		45.78		47.51		48.18		47.65		50.85	
37	51.78		48.71		47.32		47.12		50.85		47.01	
38	47.92		52.82		46.85		50.75		46.68		45.82	
39	53.08		54.32		49.38		46.58		45.65		48.41	
40	54.39		50.29		39.65		43.94		48.58		48.22	
41	50.72		49.08		45.58		45.11		48.72		53.02	
42	49.05		51.28		48.38		45.85		52.58		48.98	
43	51.75		52.89		46.05		51.75		48.78		45.71	
44	52.72		55.39		51.70		47.78		45.72		45.21	
45	54.65		53.15		48.52		43.31		45.27		48.48	
46	45.68		43.75		42.32		43.68		48.55		45.08	
47	51.49		54.75		44.41		47.91		45.01		47.45	
48	54.42		53.35		47.87		48.05		48.32		49.82	
49	53.49		51.19		46.33		42.54		49.12		48.38	
50	51.48		48.98		43.86		45.22		48.28		49.31	
51	48.38		50.22		44.06		44.94		49.28		51.19	
52	50.52		50.18		44.94		42.24		51.15		46.94	
53	50.02		47.92		44.07		58.56		46.71		46.02	
54	47.64		49.95		57.82		49.32		46.02		45.94	
55	50.79		49.18		49.37		51.41		45.98		48.35	
56	48.91		48.25		51.65		49.72		48.58		54.56	
57	48.12		47.71		49.78		47.52		54.72		53.22	
58	47.88		50.75		48.52		45.74		52.85		46.21	
59	50.50		55.32		44.62		47.05		46.35		48.18	
60	55.24		50.94		46.10		47.48		48.31		49.25	

表 2.7　第 3 回実験の各被験者実験結果（その 2）

製品数	ペア3-1				ペア3-2				ペア3-3			
	被験者3-1-1		被験者3-1-2		被験者3-2-1		被験者3-2-2		被験者3-3-1		被験者3-3-2	
	クルマ(組立)		クルマ(分解)		クルマ(組立)		クルマ(分解)		クルマ(組立)		クルマ(分解)	
	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無	時間 [s]	異常有無
61	47.80		41.44		46.08		47.15		49.22		50.08	
62	44.61		45.06		48.76		47.94		49.98		45.98	
63	44.84		49.50		49.05		47.05		46.02		49.22	
64	49.62		49.07		44.23		47.98		49.38		48.28	
65	49.55		47.21		48.95		46.68		48.11		55.35	
66	46.68		47.91		47.05		43.85		55.29		50.26	
67	48.88		46.13		44.34		44.14		50.48		46.61	
68	46.43		58.51		43.11		47.11		46.75		46.18	
69	58.04		50.72		48.95		47.42		45.88		45.98	
70	51.92		43.33		46.80		47.34		45.95		43.47	
71					47.33		47.59		43.34		45.05	
72					48.38		50.95		45.48		45.48	
73					50.47		49.91		45.51		46.38	
74					49.91		51.15		46.91		49.51	
75					51.07		47.95		49.89		37.64	
76					43.76		46.45					
77					48.15		47.15					
78					46.74		47.38					
79					46.99		45.48					
80					48.94		45.07					

製造個数	70		80		75	
合計異常回数	0		0		0	
合計作業時間 [s]	3684.90	3670.95	3800.76	3799.86	3703.30	3694.36
平均サイクルタイム [s]	52.64	52.44	47.51	47.50	49.38	49.26
データ数	18271	17838	18915	18275	18436	18236
平均データ収集頻度	4.9 データ/s（センサー3個使用のため1個あたりは1.6 データ/s）					
平均動画分析所要時間	27 s/個					

表 2.8　各実験における各被験者の作業内容計測結果　(単位 s)

(a) 第 1 回実験の計測結果

製品種類	被験者1-1		被験者1-2		被験者1-3		被験者1-4		被験者1-5	
	平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間
クルマ組立	98.52	79.18	58.38	46.61	49.03	35.15	64.76	52.02	75.51	61.06
ペンギン組立	62.71	48.61	39.50	29.18	30.84	19.65	53.19	41.93	44.07	32.78
ハウス組立	98.35	83.97	61.59	51.45	55.06	44.41	91.45	79.22	90.11	75.48

(b) 第 2 回実験の計測結果

被験者2-1		被験者2-2		被験者2-3		被験者2-4		被験者2-5		被験者2-6	
クルマ(組立)						クルマ(分解)					
平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間
69.32	54.55	54.14	44.90	55.48	43.67	44.51	33.08	44.55	33.17	42.95	29.95

(c) 第 3 回実験の計測結果

ペア3-1				ペア3-2				ペア3-3			
被験者3-1-1		被験者3-1-2		被験者3-2-1		被験者3-2-2		被験者3-3-1		被験者3-3-2	
クルマ(組立)		クルマ(分解)		クルマ(組立)		クルマ(分解)		クルマ(組立)		クルマ(分解)	
平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間	平均サイクルタイム	主作業平均時間
52.64	39.30	52.44	37.94	47.51	33.02	47.50	35.28	49.38	35.08	49.26	34.09

第 3 章　提案システム

本章では，提案するリアルタイム作業者動作分析システムについて述べる．はじめにシステムの全体像について示した後に，作業者動作データの分析部の各処理プロセスついて示す．

3.1.　システム全体像

本システムは製造現場フィールドにてウェアラブルセンサーを着用した作業者の動作データをデータベースに収集し，収集した作業者動作データの分析によりリアルタイムに製造進捗と作業履歴や主作業時間を計測することで，生産効率の改善作業や日々の製造における迅速なフィードバックを容易に実施可能にすることを狙ったシステムである．

システムの全体構成を図 3.1 に示す．本システムはフィールド，データ分析部，データ応用部の 3 要素から構成されている．フィールドは，製造現場において製品を製造する作業者の動作をウェアラブルセンサーで計測してデータベースに収集する要素である．データ分析部は，収集した作業者動作データをデータベースから取り出して分析することで，製造進捗と作業履歴や主作業時間を導出する要素である．データ応用部は，導出した製造進捗と作業履歴のデータを製造現場のシミュレーションや最適化アルゴリズムなどに入力して活用することで，さらに有用な結果を得るための要素である．

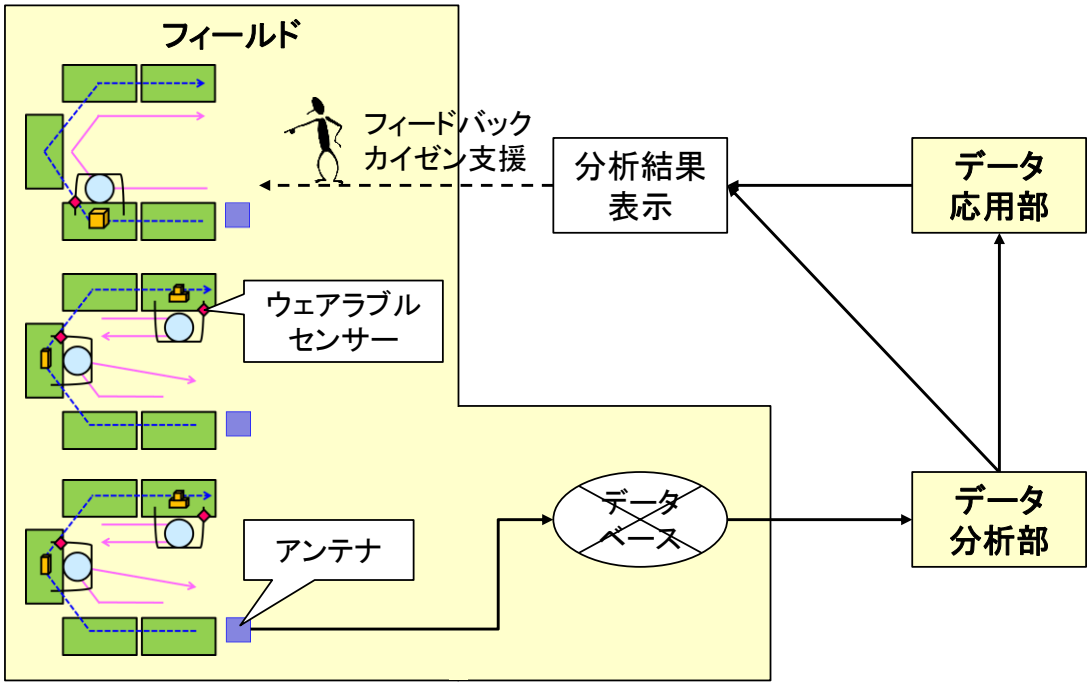


図 3.1　システム全体像

3.2. フィールド

本研究におけるフィールドは第 2 章にて設計・定義したものが一例である．机を U 字型に並べた組立セル製造ラインにおいて，ウェアラブルセンサーの StickNFind を胸・腰・足首に装着した作業者の，レゴブロックの製品製造時の動作データが，データベースに入力されていく．

なお，本研究では，実験で収集した作業者動作データをあらかじめ読み込んでから収集タイミングどおりに出力していく仮想製造実験アプリを作成することで，実験時以外でもシステムの稼働テストを行えるようにした．

3.3. データ分析部

データ分析部の処理フローを図 3.2 に示す. データ分析部はデータベースからデータを読み込むことと並行して 2 種類のデータ分析を行う．データ分析の 1 つは，ウェアラブルセンサーから得られる Beacon の RSSI データから作業者の位置を推測することで，製造進捗である製品 1 個単位のサイクルタイムを計測するものである．もう 1 つは，ウェアラブルセンサーから得られる 3 軸加速度データを分析して作業者の移動と停止の動作推定を行うことで，製品 1 個を製造する時の移動回数から作業履歴である異常動作の有無を計測する機能と，製品 1 個を製造する時の主作業時間を計測する機能である．移動回数から作業履歴である異常動作の有無を計測する機能は，製品の品質低下を引き起こす部品落下や作業忘れが発生したときに部品を拾ったり作業の後戻りをしたりという動作が増えること着目した計測である．

データ分析部は JAVA のマルチスレッドを利用した並列処理を用いることで, データベースの更新有無を確認しつつ，読み込んだデータの分析を並列して行うことを可能としている．また，データ分析が終了する前に作業者動作データが更新された時に，分析処理の競合状態が発生しないようにバイナリセマフォを用いている．図 3.2 内の変数 F, d, r, a はそれぞれ対応したスレッドをスリープさせるか実行するかを決定するためのフラグとなる変数である．

ここで，処理の流れについて説明する．メインスレッドが開始すると，まず全体フラグ F をシステム稼働中を表す 1 に変更してから，データ読込スレッド，Beacon 分析スレッド，加速度分析スレッド，その他シミュレーションなど組込応用処理スレッドを構築する．構築された各スレッドは全体フラグ F が 1 の間は動作可能になるが, 基本スリープしており，処理が必要な時に対応したフラグを通じて実行されるように制御されている．

データ読込スレッドは，Beacon 分析スレッドが実行中でなければ一定の時間間隔でデータベースが更新されているか確認し，更新されていれば，直前のサイクルタイム判定の少し前のデータから最新のデータまでを内部に格納して他の分析スレッドが参照できるようにする役割をもつスレッドである．Beacon 分析スレッドの実行中は，データ更新による分析処理の競合を防ぐためにスリープ状態にする．

Beacon 分析スレッドは，データ読込スレッドが作業者動作データを新たに格納した時に実行されるスレッドで，格納データを受け取ってから Beacon の RSSI 部分を分析して作業者の位置を推定することで製造進捗である製品 1 個単位のサイクルタイムを計測するスレッドである．分析により製品 1 個分のサイクルタイムを計測した時は，データ読み込みスレッドのフラグ d を 0 に，加速度分析スレッドのフラグ r を 1 にして，分析した作業者動作データと計測結果を加速度分析スレッドに受け渡す役割も持つ. Beacon の RSSI 分析の詳細は第 4 章で示す.

加速度分析スレッドは，Beacon 分析スレッドが新たにサイクルタイムを計測した時に実行されるスレッドである．Beacon 分析スレッドから作業者動作データと計測結果を受け取ってから，作業者動作データの加速度部分から作業者の移動と停止の回数を分析することで，製品 1 個が製造される間の異常動作の有無と主作業時間を推定するスレッドである．また，組込応用処理スレッドが存在する場合は，分析終了後に組込応用処理スレッドのフラグ a を 1 にして，分析した作業者動作データや計測結果など必要なデータを組込応用処理スレッドに受け渡す役割も持つ．加速度分析の詳細は第 5 章に示す.

外部からメインスレッド終了の命令が入ると，全体処理フラグの F が 0 に変更されるため各スレッドが終了処理に入り，すべてのスレッドが終了した後に結果をアウトプット形式にまとめてシステムは停止する．

3.4. データ応用部

データ分析部が導出する製造進捗や作業履歴である異常有無の結果は，製造現場の現状を明らかにする情報として有用であるが，製造現場への迅速なフィードバックや改善支援に繋げるためには予測や最適化と組み合わせることが必要になる．データ応用部は，データ分析部にて導出した製造進捗や作業履歴のデータを製造現場のシミュレーションや最適化アルゴリズムなどに入力して活用することを可能とする．

データ応用部の実現方法は 2 種類ある．1 つは，データ分析部の組込応用処理スレッドに処理を組み込んで連続して処理させる方法である．この方法はデータ分析部から連動するため作業者動作データや計測結果などの入力データの受け渡しが容易で，応用処理も連続的に行えるが，処理内容をコーディングしてシステムに組み込まなければならない．もう 1 つの方法としては，応用処理をさせる外部アプリを準備し，データ分析部から分析結果を json や csv などの外部アプリの入力にあった形式で出力することで，外部アプリが計測結果を用いて処理できるようにする方法が考えられる．この方法は，エージェント・シミュレーションや最適化アルゴリズムなどで既存の外部アプリを活用することができるが，データ分析部の出力形式を外部アプリにあわせて調整する必要がある．データ応用部の使用例は第 6 章で示す.

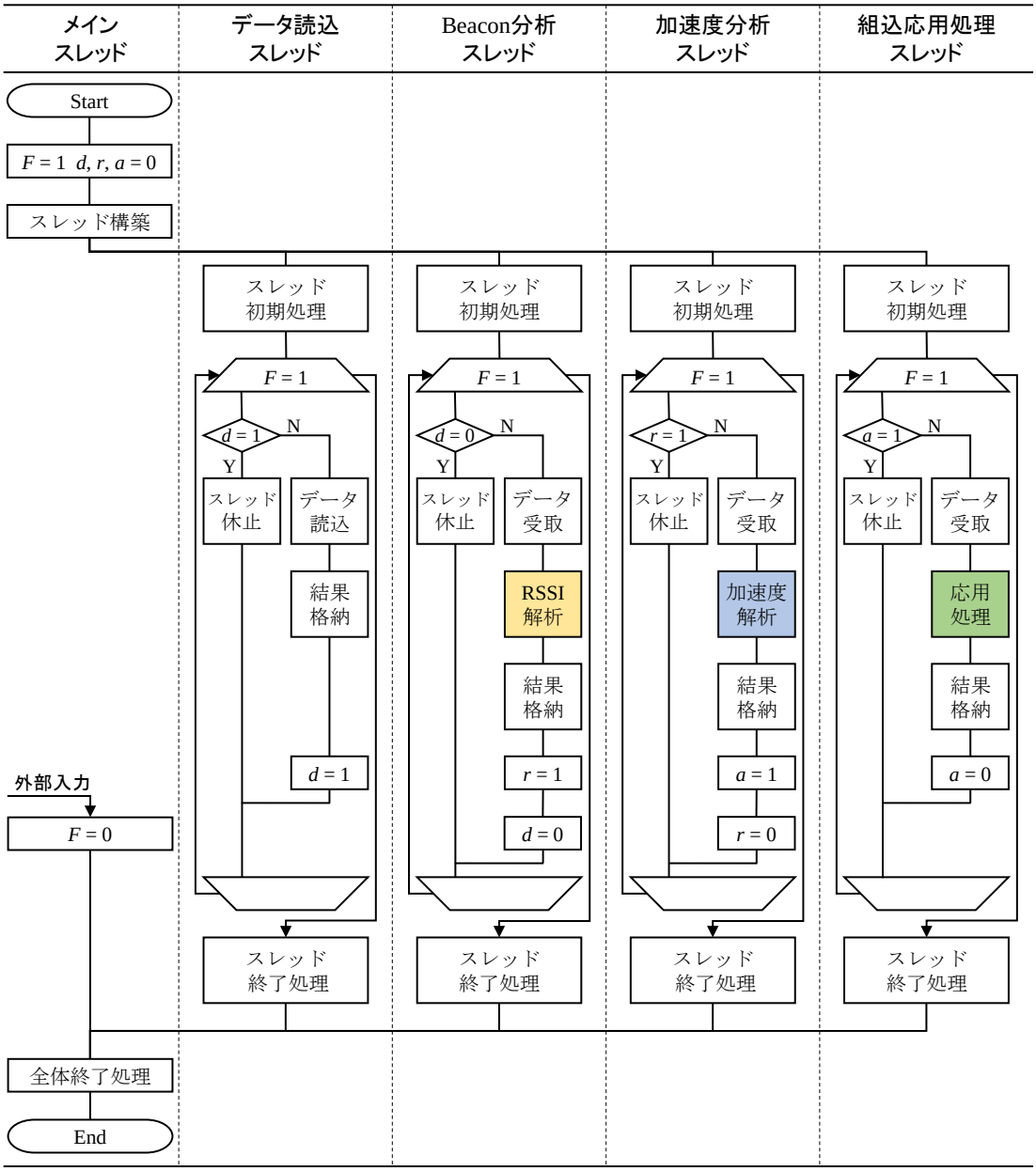


図 3.1　データ分析部の処理プロセス

第 4 章　作業者位置推定による製造進捗の導出

本章では，提案システムのデータ分析部で実行される処理の 1 つである，作業者動作データから作業者の位置を推定することにより，製造進捗である製品 1 個単位のサイクルタイムの導出について述べる．作業者動作データの Beacon の電波受信強度（Received signal strength indicator : RSSI）は，分析によりウェアラブルセンサーを装着した作業者の位置を推定することが可能である．はじめに RSSI の性質と位置推定の関連研究について示す．つぎに目標とする分析精度について述べた後に，提案システムにおける RSSI データの前処理や分析手法の実装結果について示す．なお，本研究では実験で収集した作業者動作データのうち，第 1 回実験のデータで前処理や分析手法やパラメタについて探索し，第 2 回と第 3 回実験のデータを外部データとして手法の確認に用いた．

4.1.　Beacon の電波受信強度の性質と位置推定の関連研究

電波受信強度，RSSI は無線通信において，端末が受信している電波の強度を示す指標である．単位は dBm で，1 mW のときの受信強度を 0 dBm とする相対数値で表す．Beacon から送信された電波は距離に反比例する性質をもち，RSSI は理論的にはフリスの伝達公式から以下のように計算され，図 4.1 のような関係性になる；

$$\mathrm{RSSI} = \mathrm{TxPower} - 20 \times \log_{10} d \qquad (4.1)$$

ここで, TxPower はセンサーとアンテナ間の距離を 1 m にしたときの実測値を表し運用時は事前にキャリブレーションとして計測しておいた値を用いる事が多い変数で，d は実際のセンサーとアンテナ間の距離を表す変数である．

式(4.1)を利用することで，あるアンテナ配置でセンサーが移動した時に RSSI がどのような推移をするか計算したり，実測した RSSI からセンサーとアンテナ間の距離を推測したりすることができる. [Apple2014]では, 計測した RSSI によってセンサー装着者との距離を 0-1 m

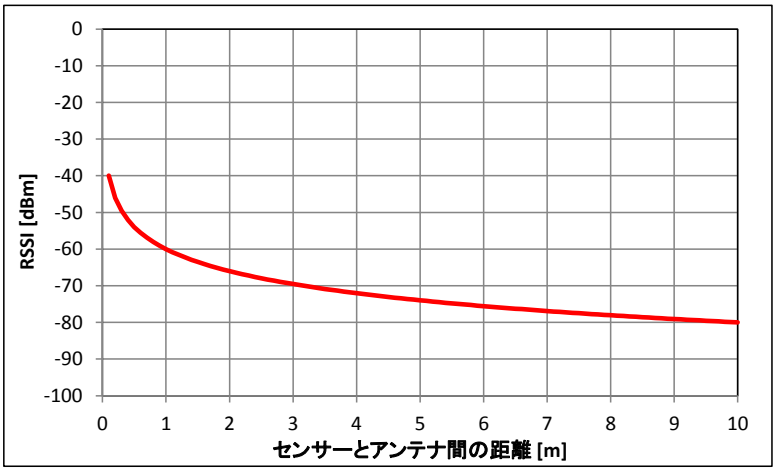


図 4.1　RSSI と距離の関係（TxPower = -60 とした場合）

圏内，1-3 m 圏内，3 m 以上の 3 分類で推定している．

実際の RSSI では Beacon の電波が床や天井で反射や吸収を起こして誤差が乗りやすいことと，アンテナとセンサーの方向関係までは分からないことから，複数のアンテナでセンサーからの電波を計測することで，センサー所持者の位置を推定する方法も提案されている．[佐藤 2011]では，RSSI が場所の影響を受けて計測値が変わることを考慮して様々な場所でセンサーとアンテナの距離と電波受信強度の関係を実測し，複数のアンテナの位置と RSSI から得られる距離を用いることでセンサー所持者の存在範囲を絞り込めることを示した．[川村 2011]では，ある距離における Wi-Fi の RSSI の条件付き確率密度関数の式を導出しておき，RSSI からセンサーと複数のアンテナとの距離を最尤推定法により推定し，位置推定を行った．[春本 2011]では，アンテナが疎である状況でも位置推定の精度を確保するために，条件付き確率密度関数に加えて存在確率分布を求める手法を用いた．[Bahl2000]，[Chen2012]，[坂 2014]では，実環境のフィールドと RSSI のデータベースをあらかじめ作成しておき，実測された RSSI とデータベースの値を参照することで位置推定を行う Fingerprint と呼ばれる手法を用いた．また，[北澤 2010]，[藤野 2013]，[大畠 2013]では，複数のアンテナをフィールド内の交差路に設置し，センサー所持者が通過した交差点の順番を明らかにすることで，位置を推定する手法を用いた．しかしながら，これらの手法による位置測定は誤差が大きく，組立セル製造ラインを対象とする場合，製造ラインのサイズを考慮すると製造中の作業者の位置を常に把握することは難しい．

そこで本研究では，組立セル製造ラインでは作業位置や順序が定められているため，図 4.2 のようにアンテナを作業開始位置付近に置くことで，RSSI が図 4.3 のような製品 1 個の製造開始時にピークを持つ周期的な波形にできることを利用した．RSSI のピーク抽出により作業者が製品 1 個の製造を開始した時刻を特定することで，製造進捗である製品 1 個のサイクルタイムの導出が可能となる．なお，図 4.2 と図 4.3 は 1 人用セルにおいてクルマ組立を製造するときを示しているが，2 人協力用セルでも前半作業者と後半作業者の作業開始位置付近にアンテナを置くことで，RSSI は同様の周期性を示す．

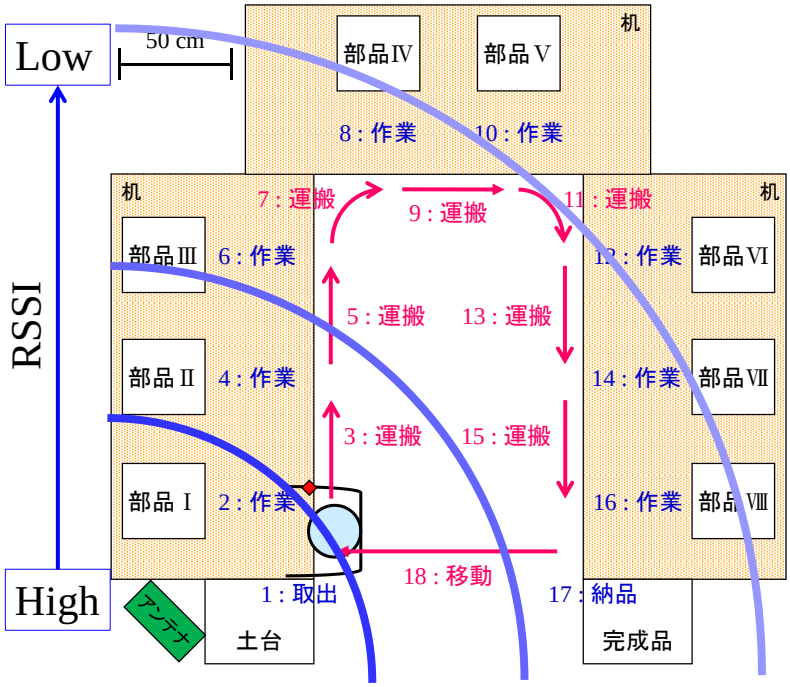


図 4.2　1 人用セルにおけるアンテナ配置

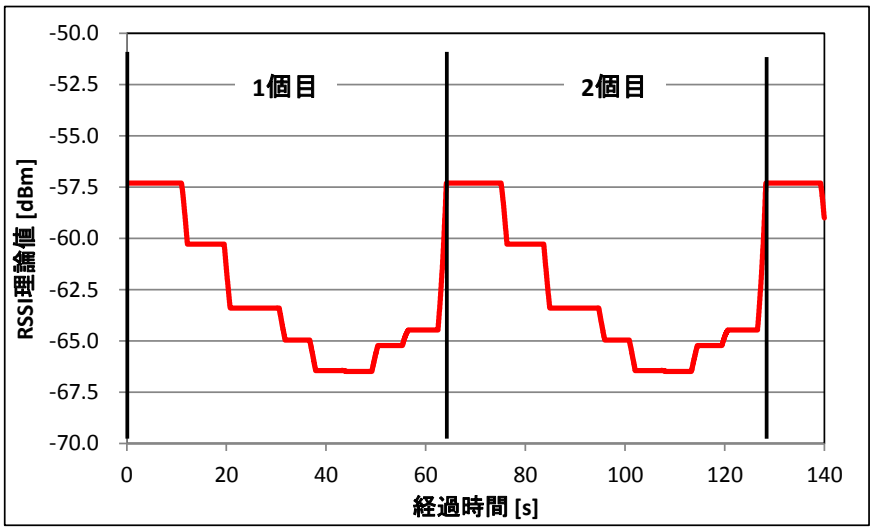


図 4.3　クルマ組立製品製造における標準作業時の RSSI 理論値の推移

4.2. 本章の目標

RSSI が理論値どおりに計測できる場合は製造開始位置を示すピークを精度よく抽出できるが，実際には周辺環境の影響や作業者の作業時の揺れもあり，ノイズが乗った値が計測される．そこで，標準作業時の RSSI の理論値に平均 0 で標準偏差を 0-8 まで 1 刻みに変化する正規分布に従う乱数をノイズとして乗せた時の，サイクルタイム仮想値の標準サイクルタイムからの平均絶対誤差をシミュレーションで求めた（図 4.4）．サイクルタイム仮想値は，製品 102 個を連続製造したと想定した場合の計算結果から中 100 個分の値を抽出して求めた．また，各製品における製造開始のタイミングは，標準サイクルタイム内でノイズを乗せた RSSI 計算値が最も高くなるタイミングとした．

各標準偏差について 10 回ずつシミュレーションした時の結果を図 4.5 に示す．ノイズが増えるにつれて，平均絶対誤差も増加する傾向が見える．ここで，各被験者の製造前に計測した Beacon の RSSI の標準偏差は，第 1 回実験で 5.1，第 2 回実験で 5.7，第 3 回実験で 6.3 であった．したがって図 4.5 より，第 1 回実験の作業者動作データを使う RSSI 分析のデータ前処理や分析手法やパラメタ探索では平均絶対誤差 2.1-2.7s を目標に，第 2 回と第 3 回実験の作業者動作データへの適用では平均絶対誤差はそれぞれ 2.5-3.4s, 3.2-4.1s を目標とする．

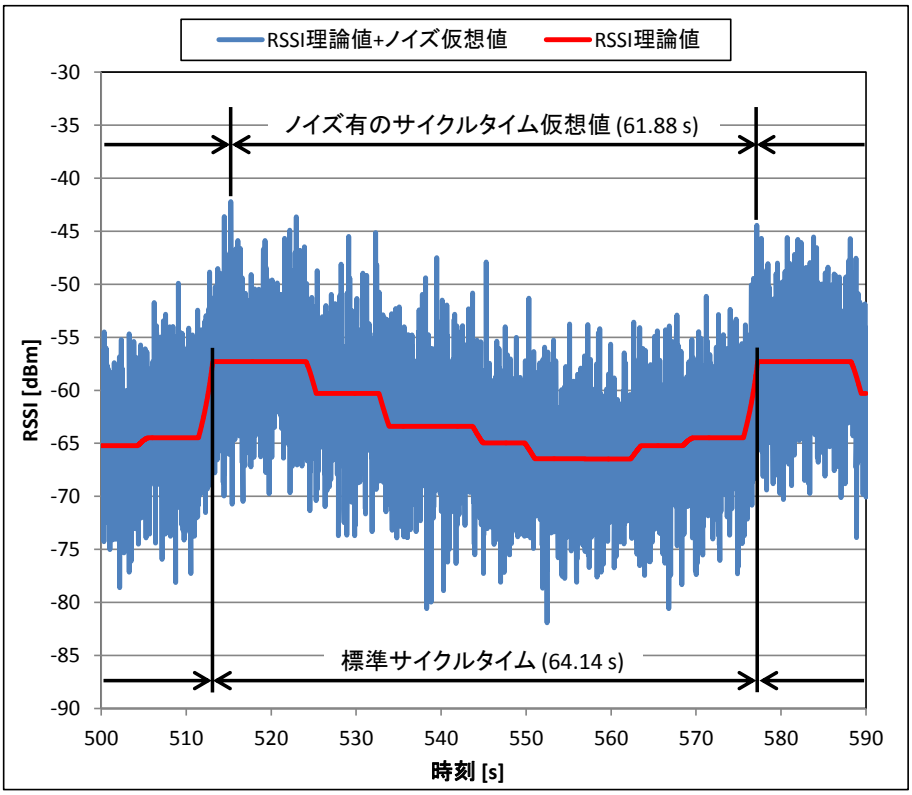


図 4.4　仮想サイクルタイムの導出（平均 0，標準偏差 5 の正規乱数での例）

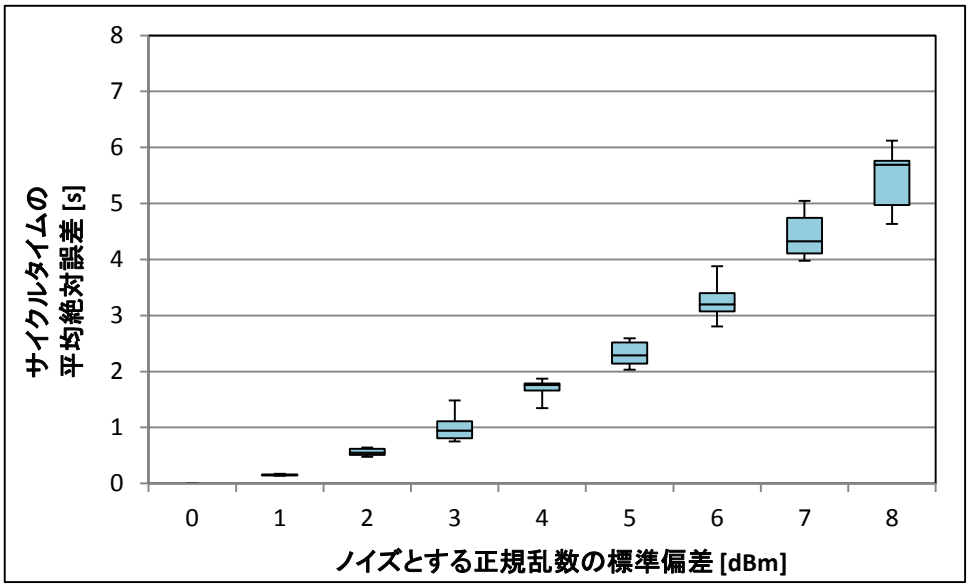


図 4.5　ノイズとする正規乱数の標準偏差とサイクルタイムの平均絶対誤差の関係

4.3. RSSI データ分析手法

4.3.1 データ前処理手法とピーク抽出手法

データ前処理とピーク抽出手法およびそのパラメタの探索により，第 1 回実験の作業者動作データの RSSI を分析時に，目標とする分析精度を満たせる条件を求める．

データ前処理としてのノイズ除去では，Moving average filter（以下，MAF），Savitzky-Golay filter（以下，SGF），ノイズ除去無しの 3 手法を比較する[Orfanidis2010]．MAF は処理対象のデータ 1 個を，前後範囲 m 個を含む 2m+1 個のデータの平均値に置き換えることで，ノイズを除去する手法である．一方で SGF は処理対象のデータ 1 個を，前後範囲 n 個を含む 2n+1 個のデータに対して指定した次数 o の多項式を最小二乗適合により導出してから，処理対象データの部分の多項式の値に置き換える手法である．SGF は高周波成分をある程度保存したままノイズ除去ができるというメリットがあるが，MAF よりもノイズ除去性能は低いというデメリットもある．

ピーク抽出手法では，閾値を設定する方法とε-tube 法の 2 種類を比較する．閾値設定は，ある閾値を越えたデータ群のうち，最も数値の高い値もしくは低い値をピークとして抽出する方法である（図 4.6）．ε-tube 法はあるデータ 1 個を処理する場合に，そのデータより過去 k 個のデータで平均値を計算してから，別に定めるεを用いて Epsilon tube とよばれる移動

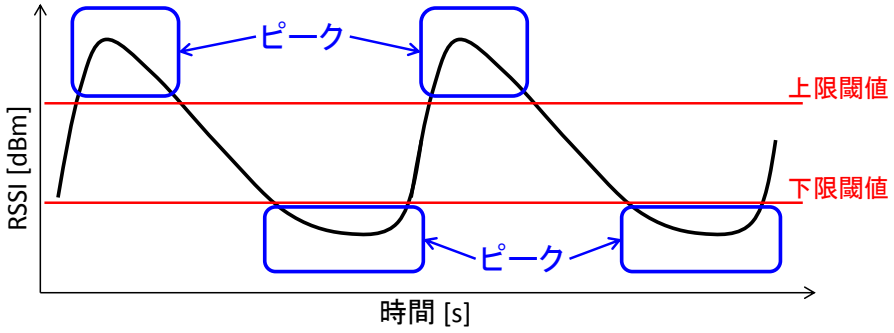


図 4.6 閾値設定によるピーク抽出

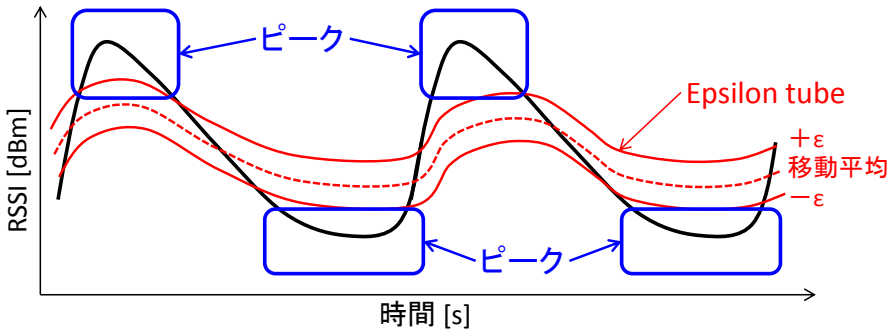


図 4.7 Epsilon tube によるピーク抽出

表 4.1　データ前処理とピーク抽出手法のパラメタ水準

項目		手法	変数名	パラメタ水準
Beacon 分析	前処理	無し		
		Moving average filter	範囲 m	5，10
		Savitzky-Golay filter	範囲 n 次数 o	5，10 2
	抽出	閾値	閾値	－40 ～ －70　－2 刻み
		ε-tube	範囲 k ε	10，20 1 ～ 10　1 刻み

平均±εの領域を設定し，該当データが Epsilon tube 外に出てから再び Epsilon tube 内に戻るまでの波形をピーク波形として抽出する手法である．フーリエ変換やウェーブレット変換よりも計算が容易という特徴がある（図 4.7）[村尾 2010]，[村尾 2011]．本研究では，製品の製造開始は RSSI が最も大きくなる時のため，抽出したピークのうち，上側ピークのみを用いて分析を行った．

これらのデータ前処理 3 種類とピーク抽出 2 種類を組み合わせた 6 種類に，各手法でのパラメタ水準を表 4.1 のとおりに設定して，分析条件とした．

4.3.2　第 1 回実験データによる条件探索

第 1 回実験の製品 30 個×5 名分の作業者動作データに，設定した全ての分析条件を適用した結果を表 4.2 に示す．パラメタを含めると条件数が多くなるため，各手法の組み合わせで最もよくなったパラメタの結果のみ示している．ここで，表 4.2 の[ ]内の数字は各手法のパラメタを表し，製造数計測精度は製品合計 150 個のうち何%の製品のサイクルタイムをデ

表 4.2　第 1 回実験データを用いた条件探索結果

条件番号	RSSI 分析		製造数計測精度 [%]	サイクルタイムの平均絶対誤差 [s]
	前処理	抽出		
1	無し	閾値 [-50]	92.7	3.0
2	MAF [10]	閾値 [-68]	94.7	2.1
3	SGF [10, 2]	閾値 [-66]	98.7	3.0
4	無し	ε-tube 法 [10, 5]	0.0	---
5	MAF [10]	ε-tube 法 [20, 3]	96.0	2.1
6	SGF [10, 2]	ε-tube 法 [20, 8]	79.3	6.4

ータ分析により計測できたかを示し，サイクルタイムの平均絶対誤差はデータ分析で計測できたサイクルタイムを対象として計算した値である．

結果より，条件 1, 2, 3, 5 の 4 つは，製造数計測精度も 90 %以上で，平均絶対誤差も 2.1-3.0 s と目標分析精度に近い結果となった．そこで，この 4 条件を外部データである第 2 回と第 3 回実験の作業者動作データに適用して，精度の変化を確認する．

4.3.3　第 2 回と第 3 回実験データへの適用結果

条件番号 1, 2, 3, 5 の 4 つを，第 2 回と第 3 回実験の作業者動作データのうち 50 個×6 名分に適用した結果を表 4.3，表 4.4，表 4.5，図 4.8 に示す．第 1 回実験データへ適用したときと同様に，サイクルタイムの平均絶対誤差はデータ分析で計測できたサイクルタイムを対象として計算した値である．なお，表 4.5 でサイクルタイムが記載されていない被験者は，該当の分析条件の時に測定できたサイクルタイムがなかったことを示す．また，図 4.8 は条件番号 5 を適用した時の，第 2 回実験における被験者 1 名の目視分析と本システムそれぞれによる分析の時系列結果であり，前後 5 データずつを用いた移動平均と微分値も同時に記載している．

表 4.3　第 2 回実験データへの条件適用結果

条件番号	RSSI 分析		製造数計測精度 [%]	サイクルタイムの平均絶対誤差 [s]
	前処理	抽出		
1	無し	閾値 [-50]	54.3	3.5
2	MAF [10]	閾値 [-68]	95.0	2.5
3	SGF [10, 2]	閾値 [-66]	97.7	2.5
5	MAF [10]	ε-tube 法 [20, 3]	90.0	2.5

表 4.4　第 3 回実験データへの条件適用結果

条件番号	RSSI 分析		製造数計測精度 [%]	サイクルタイムの平均絶対誤差 [s]
	前処理	抽出		
1	無し	閾値 [-50]	64.3	5.0
2	MAF [10]	閾値 [-68]	11.3	4.7
3	SGF [10, 2]	閾値 [-66]	19.5	4.8
5	MAF [10]	ε-tube 法 [20, 3]	78.7	4.1

表 4.5　第 3 回実験データにおける条件適用時の個人別結果

条件番号	RSSI分析		平均サイクルタイム［s］					
	前処理	抽出	ペア1		ペア2		ペア3	
			被験者 3-1-1	被験者 3-1-2	被験者 3-2-1	被験者 3-2-2	被験者 3-3-1	被験者 3-3-2
真値			53.97	53.83	47.42	47.40	49.90	49.98
1	無し	閾値 [-50]	56.54	55.34	48.07	47.82	49.93	47.91
2	MAF [10]	閾値 [-68]	---	55.50	---	---	---	---
3	SGF [10, 2]	閾値 [-66]	---	54.17	---	48.03	45.16	51.44
5	MAF [10]	e-tube法 [20, 3]	54.53	54.58	47.58	47.01	50.29	50.51

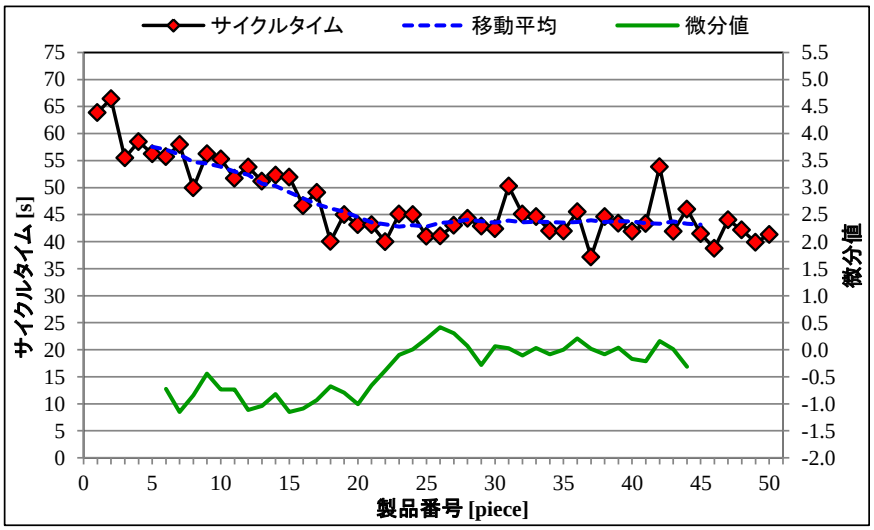
表 4.3 と表 4.4 より，条件 1 は第 2 回と第 3 回実験データの両方でサイクルタイムの平均絶対誤差は目標に達せず，製造数計測精度 54.3 %，64.3 %と低下した．また，条件 2 と条件 3 は第 2 回実験データではサイクルタイムの平均絶対誤差が目標に達したが，第 3 回実験データへの適用では製造数計測精度が 11.3 %と 19.3 %と大幅に低下し，サイクルタイムの平均絶対誤差も目標に達しなかった．結果として条件 5 のデータ前処理に MAF を，ピーク抽出にε-tube 法を用いるのが最良の結果で，第 3 回実験データへの適用で製造数計測精度の低下が見られるが，第 2 回と第 3 回実験データの両方でサイクルタイムの平均絶対誤差が目標に達した．

さらに，表 4.5 より第 3 回実験のように 1 つのセル製造ラインで複数人が同時に作業する場合でも，作業者ごとにサイクルタイムの実績を収集することができており，本システムでは個人特定が問題なくできていることが確認できた．

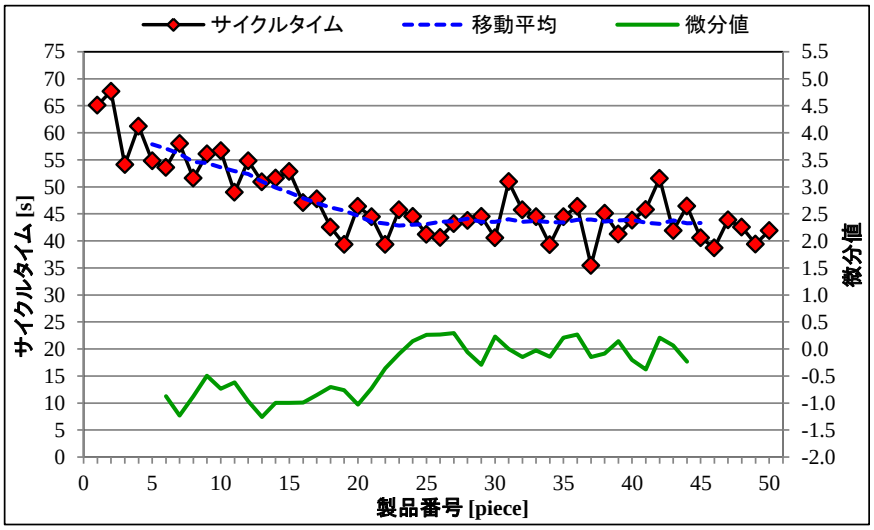
そして，条件 5 を適用した図 4.8 より，本システムの計測結果が目視分析と同様の時系列変化をしていることが確認できた．この時の目視分析による分析結果と本システムによる分析結果の相関係数は 0.97 であった．また，対象の作業者では開始から 25 個程度までサイクルタイムが低減し，25 個以降は同程度のサイクルタイムで製造を続けていることが分かった．

4.3.4　考察

データ前処理で最も良い結果を得たのは MAF であった．これはピークを抽出する RSSI の波形の周期が 1 製品の製造時間となり，本研究では標準サイクルタイムの約 60 秒で第 1 回実験では約 300 データ，第 2 回と第 3 回実験では約 100 データと十分な大きさがあるため，高周波成分を残す必要がなくなり，ノイズ除去力が高い MAF の効果が向上したと考えられる．



(a) 目視分析による分析結果



(b) 本システムによる分析結果

図 4.8　第 2 回実験における作業者 1 名の時系列結果

ピーク抽出で最も良い結果を得たのはε-tube 法であった．閾値を用いた条件では，特に第 3 回実験の作業者動作データに適用時には前処理を行っても製造数計測精度が低下した．この時の分析結果を詳細に見ると，閾値を用いた条件で製造数が正確に計測できなかったのは，閾値ではピークが抽出できず複数個の製品を分離できなかったためと分かった．これは，第 1 回実験のデータで最良となった閾値の条件が 1 人用セルの RSSI 波形に過適合しており，第 3 回実験でフィールドの形が 2 人協力用セルに変化したことによる RSSI 波形の変化に対応できなかったためと考えられる．したがって，移動平均の値を用いて動的に基準を変更していくε-tube 法が，本システムにおけるピーク抽出手法として適していたと考えられる．

図 4.8 に示した作業者のサイクルタイムの時間による変化は，作業者が製造に習熟したことによるものと考えられる．また，本研究の実験時間内では確認できなかったが，長時間にわたり作業を続けていれば疲労によるサイクルタイムの遅延も本システムにより計測できたと期待できる．長時間にわたって作業の実績値を自動的に収集し，習熟や疲労の影響も加味した活動が可能になるのは本システムの大きな利点である．さらに，本研究では実験のため 1 つのラインしか扱っていないが，多くの製造ラインがある製造現場へ本システムを適用すれば製造現場全体の製造能力を把握することができ，製造現場における作業者のデータ収集の負担軽減だけでなく，全体最適な改善に貢献できると考えられる．

4.4. まとめ

本章では，提案システムのデータ分析部で実行される処理の 1 つである，作業者動作データから作業者の位置を推定することによる，製造進捗である製品 1 個単位のサイクルタイムの導出について述べた．

Beacon の RSSI はアンテナとの距離と反比例する性質を持つことと，組立セル製造ラインでは作業位置や手順が定められているためアンテナを作業開始位置付近に置くと RSSI が製品 1 個の製造開始時にピークを持つ周期的な波形になることを利用して，RSSI のピーク抽出により作業者位置推定を行い，製品 1 個の製造開始時刻の特定により製造進捗である製品 1 個単位のサイクルタイムを導出した．分析はデータ前処理に無し・MAF・SGF の 3 種類，ピーク抽出に閾値設定とε-tube 法の 2 種類を組み合わせた手法で行い，第 1 回実験の作業者動作データを用いて分析の条件探索をして，最も良い条件を第 2 回と第 3 回実験の作業者動作データへ展開して評価した．

結果として，データ前処理に前後 10 個のデータで移動平均を計算する MAF 手法，ピーク抽出に前 20 個のデータの平均値に±3 をした Epsilon tube を用いてピークを抽出するε-tube 法の組み合わせが最も適した条件であり，本システムにより作業者単位での分析も可能となることを明らかにした．

第 5 章　加速度分析による作業履歴と主作業時間の導出

本章では，提案システムのデータ分析部で実行される処理の 1 つである，作業者動作データから作業者の移動と停止を推定することにより，作業履歴である製造中の異常動作の有無と，主作業時間の計測について述べる．作業者動作データの作業者の腰と足首の 3 軸加速度は，分析により作業者の動作を推定することが可能である．はじめに，加速度の性質と動作推定の関連研究について示す．つぎに本章の目的について述べた後に，提案システムにおける加速度データの前処理や分析手法の実装結果について示す．なお，本研究では実験で収集した作業者動作データのうち，第 1 回実験のデータで前処理や分析手法やパラメタについて探索し，第 2 回と第 3 回実験のデータを外部データとして手法の確認に用いた．

5.1.　加速度の性質と動作推定の関連研究

加速度とは単位時間あたりの速度の変化割合を示す値で，単位は m/s^2 で表す．加速度センサーを作業者の身体各所に装着することで，装着部それぞれの動きを定量データとして収集することができるため，収集した加速度データを分析することで作業者動作の推定を狙うことができる．

加速度により動作を推定する手法は，これまでに多く提案されてきている．[中川 2016]では身体各部につけた加速度センサーから加速度の微分値である躍度を計算し，閾値を設定することで，避難行動時の移動・停止や首を振ったか否かといった動作の特定を可能にした．[Najafi2003]では老人の胸に取り付けたセンサーの加速度と角速度から，運動学的なモデルと Wavelet 変換を用いて立つ，歩くなどの動作の特性を把握することで，特製に基づいた各動作の閾値を設定して判定することで動作特定を行った. [小林 2006]では腰部背側に装着した加速度センサーで収集した歩行時の加速度データに対して，鉛直・水平方向成分への分解とオフセット誤差の累積を低減する算出法を適用することで歩行軌跡の導出を可能にしている．また，歩行動作を運動学的に評価する指標も提案している．[Gafurov2009]では加速度センサーを用いた歩行者認識を，歩行周期毎に分割した加速度を事前に収集したデータと比較することで行っており，さらに靴の種類による認識精度の低下や足の水平方向成分が効果的との知見を示した．[黄 2016]では頭，腰，手首，足首に加速度センサーを装着した被験者に図書館で避難訓練を行わせ，収集したデータに対して Deep neural network の一種で時系列データを扱うのに適している Long short-term memory を適用して，移動と停止や見回すといった動作の推定をしている．これらは，主に歩行中の加速度の変化に注目して分析を行っている．

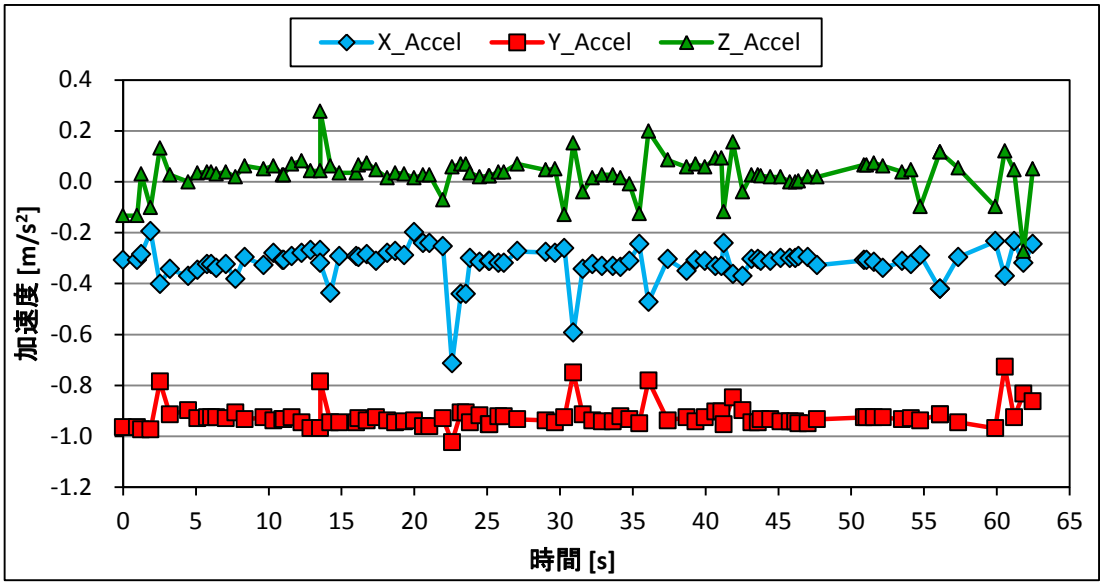


図 5.1　製造時の腰の加速度データ（第 1 回実験より製品 1 個分を抜粋）

[Wang2005]では道具に取り付けた加速度センサーのデータに対して，サポートベクターマシンの判定を行うことで人の動作の特定を行った．[Koskimaki2009]では手首に装着したセンサーからネジ止めやスパナ使用の加速度と角速度データを収集し，移動窓法と k-近接法を組み合わせることで作業の分類を行った．[Altun2010]では両手首，両膝，胸に装着した加速度センサーで 19 種の動作データを収集し，導出した最大値などの様々な特徴量に対して複数の機械学習手法を適用して動作推定の精度比較を行った．[Joshua2011]では腰に装着した加速度センサーからレンガ積作業の加速度データを収集し，単純ベイズやニューラルネットワークでモルタルの塗布やレンガ運びなどの作業分類を行った．[Tontani2013]では作業の成功パターンの波形を事前に準備しておき，実際の計測時に Dynamic time warping に基づいて判定することで動作のリズムの分析を行った．[矢田 2014]では手首に加速度センサーを含む複合ウェアラブルセンサーを装着し，収集した加速度データから 1 分あたりの腕の動く回数を計測することで，日々の動きの活発さを評価している．[村上 2014]では加速度データに対して自然言語処理分野のトピックモデルを適用することで 10 種の動作分類を行った．これらは動作の種類に注目して分類を行っている．

本研究では作業者の移動と停止を推定することで，移動回数によって作業履歴の計測と，停止中の主作業時間の計測を行う．そのため，[中川 2016]の手法を参考に腰と足首の加速度の微分値である躍度に閾値を設けて移動と停止を判定する手法と，[小林 2006]や[Gafurov2009]を参考に腰の加速度の水平成分と鉛直成分を分解して分析する手法を導入する．

5.2. 本章の目標

作業者動作データの加速度の分析により各作業者の作業履歴や作業内容を低コストで計測できるようにすることで，稼働分析による作業効率化や作業者育成に資することが本研究の目的の 1 つである．組立セル製造ラインにおける主作業時間の測定は，従来の目視計測では定常的に行うことが難しく，映像の自動分析では複数人が同時に作業する場面で個人ごとに常時計測するのは制約が厳しいという課題がある．

本章では，提案システムの加速度分析により得られる作業履歴である異常有無の精度が高い手法を探索するとともに，同様に得られる主作業時間の結果において複数人が同じ範囲で作業を行う 2 人協力用セルを用いた第 3 回実験でも被験者ごとに得られ，映像目視分析で得た真値と同様の傾向を示すか否かで評価し，稼働分析や作業者育成といった現場の活動に活用できることを示すことで提案システムの加速度分析の有用性を示す．

5.3. 加速度データ分析手法

5.3.1 データ前処理手法と加速度分析手法

データ前処理とパラメタ設定，および，腰の加速度の水平成分抽出と躍度の閾値設定により，第 1 回実験の作業者動作データの加速度を分析時に，最も良い結果を得られる条件を求める．なお，本システムにおける加速度分析は，Beacon の RSSI 分析により製品 1 個分に分割された作業者動作データに対して行われる．したがって，加速度分析の条件は Beacon の RSSI 分析の条件と組み合わせて確認する．

データ前処理としてのノイズ除去では，RSSI の分析時と同様に，Moving average filter（以下，MAF），Savitzky-Golay filter（以下，SGF），ノイズ除去無しの 3 手法を比較する．

腰の加速度の成分分解では，成分分解ありと無しの 2 手法を比較する．ここで，腰の加速度分解による鉛直方向ベクトルと水平方向ベクトルの大きさは以下の計算で求める．まず，各実験の開始前に収集したキャリブレーション用の静止時データから基準鉛直方向ベクトル$V_{pre} = \{x_{pre}, y_{pre}, z_{pre}\}$を求めておく．つぎに，ある時刻$t$に加速度ベクトル$P_t = \{x_t, y_t, z_t\}$が得られた時に，鉛直方向ベクトル$V_t$と水平方向ベクトル$H_t$は，

$$\mathrm{V}_t = \frac{\{-1 \times x_t \times x_{pre} + (-1) \times y_t \times y_{pre} + (-1) \times z_t \times z_{pre}\}}{\|V_{pre}\|} \tag{5.1}$$

$$H_t = \sqrt{\|P_t\| - \|V_t\|} \tag{5.2}$$

という式で与えられる．なお，この加速度の成分分解は腰の加速度データに対してのみ行い，足首の加速度データは加速度の大きさのみを使用する．これは，作業者が移動で足を動かすときは足に装着したウェアラブルセンサーの向きが変わり，静止時の重力による加速度ベクトルの向きが鉛直成分を示すとは言えなくなるためである．

足首の加速度の大きさ，成分分解ありの場合は腰の加速度水平成分の大きさ，成分分解

なしの場合は腰の加速度の大きさに対して，躍度を計算する．データn個目の躍度J_nは，データn個目の時刻をt_n，対象の加速度の大きさをS_nとすると，$n-1$個目のデータの値も用いて，

$$\mathrm{J}_n = \frac{S_n - S_{n-1}}{t_n - t_{n-1}} \tag{5.3}$$

で求める．この躍度の絶対値に対して閾値を設定することで，躍度が小さい停止時と躍度が大きい停止時を切り分けて，動作推定を行う．

各条件の評価は，動作推定による移動回数が標準作業で設定されている移動回数と比較して乖離しているかで行う．第 1 回と第 2 回実験では標準作業における移動回数は 8 回で，第 3 回実験では標準作業における移動回数は 7 回である．しかしながら，第 2 章の実験結果で示したとおり，本研究の実験では本来標準作業として想定していない「ベース置場前から部品 I 置場前への運搬」と「部品 VIII 置場前から完成品置場前への運搬」の 2 回が標準作業に追加して発生することがあった．さらに，本研究の標準作業はベース取り出し時を製品の製造開始と設計しているが，RSSI 分析による製造開始タイミングの判定ズレによりベース取り出し前の移動 1 回を追加して計測する可能性がある．したがって，本研究では第 1 回と第 2 回実験では移動 8~11 回を，第 3 回実験では移動 7~10 回を異常なしとして評価する．

5.3.2　第 1 回実験データによる条件探索

それぞれのデータ前処理手法とピーク抽出手法で設定するパラメタ水準を表 5.1 に示す．これらの手法とパラメタを組み合わせて，第 1 回実験の製品 30 個×5 名分の作業者動作データに適用した結果を表 5.2 に示す．ここで，表 5.2 の[ ]内の数字は各手法のパラメタを表し，動作異常計測精度は Beacon の RSSI 分析でサイクルタイムを分割できた製品のうち何%の製品の動作異常有無を真値と同じ結果に評価できたかを示している．また，F-measure は

表 5.1　データ前処理と加速度分析手法のパラメタ水準

項目		手法	変数名	パラメタ水準
加速度分析	前処理	無し		
		Moving average filter	範囲 m'	5，10
		Savitzky-Golay filter	範囲 n' 次数 o'	5，10 2
	成分分解	鉛直・水平成分分解		無，有
	判定	閾値	腰 足首	0.01 ～ 0.20　0.1 刻み 0.01 ～ 0.20　0.1 刻み

表 5.2　第 1 回実験データを用いた条件探索結果

条件	RSSI分析		加速度分析			製造数計測精度［%］	作業履歴計測精度［%］	F-measure
	前処理	判定	前処理	分解	判定			
1	無し	閾値 [-50]	SGF [10, 2]	[有り]	腰 [0.01] 足 [0.12]	92.7	69.8	0.782
2	MAF [10]	閾値 [-68]	SGF [10, 2]	[有り]	腰 [0.01] 足 [0.16]	94.7	66.2	0.755
3	SGF [21, 2]	閾値 [-60]	SGF [10, 2]	[有り]	腰 [0.01] 足 [0.12]	98.7	65.8	0.751
4	無し	ε-tube法 [10, 5]	全手法・パラメタで同結果			0.0	0.0	0.000
5	MAF [10]	ε-tube法 [20, 3]	SGF [10, 2]	[有り]	腰 [0.01] 足 [0.12]	96.0	66.7	0.757
6	SGF [21, 2]	ε-tube法 [20, 8]	無し	[有り]	腰 [0.01] 足 [0.12]	79.3	68.9	0.777

分類問題の精度評価でよく使われる値で 1 に近いほど性能が良いことを示す．なお，パラメタを含めると条件数が多くなるため，各 Beacon の RSSI 分析条件に対して最も精度のよくなった手法とパラメタを組み合わせた結果のみ示している．

結果より，データ前処理の SGF と，腰の加速度の成分分解有りを組み合わせた手法を用いるのが，Beacon の RSSI 分析で計測精度の高かった条件 1, 2, 3, 5 で，動作異常計測精度が 65%-70%，F-measure も 0.75-0.80 と同程度の結果となった．そこで，Beacon の RSSI 分析の精度に合わせて条件 1, 2, 3, 5 の 4 つを選択し，外部データである第 2 回と第 3 回実験の作業者動作データに適用して精度の変化を確認する．

5.3.3　第 2 回と第 3 回実験データへの適用と考察

条件 1, 2, 3, 5 の 4 つを，第 2 回と第 3 回実験の作業者動作データのうち 50 個×6 名分に適用した結果を表 5.3 と表 5.4 に示す．結果より，条件 1, 2, 3, 5 のいずれも，第 2 回と第 3 回実験の両方の作業者動作データに対して，動作異常計測精度で 65%程度，F-measure で 0.8 程度の精度を得られた．Beacon の RSSI 分析の条件もあわせて考えると，加速度に対してデータ前処理に SGF，腰の加速度成分の分解をありにして，腰と足首の躍度の閾値をそれぞれ 0.01，0.12 に設定する条件 5 が最も適していると考えられる．

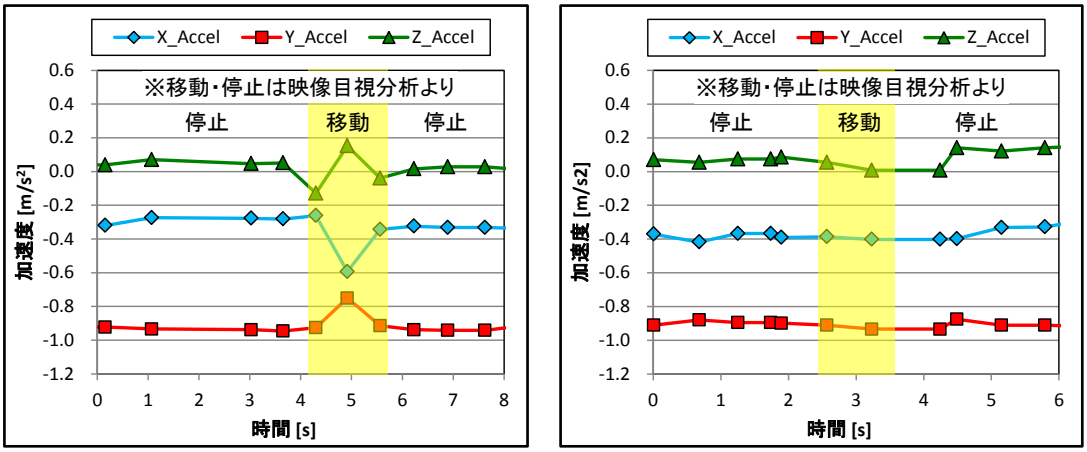
データ前処理に SGF が適していた理由は，セル製造ラインにおける移動は 1 秒程度の短時間で数データ程度の短いピークが立つ波形になるため，高周波成分を残しながらノイズを除去する手法が有効になったと考えられる．しかしながら，移動判定箇所を詳細に確認したところ，失敗箇所では移動中にもかかわらず加速度が変化していない部分が見られた．これは加速度データのサンプリングレートが十分でなかった影響と考えられる．本システムの活用時は加速度データのサンプリングレートを大きくすることが重要である．

表 5.3　第 2 回実験データへの条件適用結果

条件	RSSI分析		加速度分析			製造数計測精度［%］	作業履歴計測精度［%］	F-measure
	前処理	判定	前処理	分解	判定			
1	無し	閾値 [-50]	SGF [10, 2]	[有り]	腰 [0.01] 足 [0.12]	54.3	60.7	0.754
2	MAF [10]	閾値 [-68]	SGF [10, 2]	[有り]	腰 [0.01] 足 [0.16]	95.0	67.4	0.794
3	SGF [21, 2]	閾値 [-60]	SGF [10, 2]	[有り]	腰 [0.01] 足 [0.12]	97.7	65.9	0.788
5	MAF [10]	ε-tube法 [20, 3]	SGF [10, 2]	[有り]	腰 [0.01] 足 [0.12]	90.0	67.4	0.798

表 5.4　第 3 回実験データへの条件適用結果

条件	RSSI分析		加速度分析			製造数計測精度［%］	作業履歴計測精度［%］	F-measure
	前処理	判定	前処理	分解	判定			
1	無し	閾値 [-50]	SGF [10, 2]	[有り]	腰 [0.01] 足 [0.12]	64.3	70.5	0.827
2	MAF [10]	閾値 [-68]	SGF [10, 2]	[有り]	腰 [0.01] 足 [0.16]	11.3	64.7	0.786
3	SGF [21, 2]	閾値 [-60]	SGF [10, 2]	[有り]	腰 [0.01] 足 [0.12]	19.5	55.9	0.717
5	MAF [10]	ε-tube法 [20, 3]	SGF [10, 2]	[有り]	腰 [0.01] 足 [0.12]	78.7	67.4	0.801



(a) 移動判定の成功箇所　　(b) 移動判定の失敗箇所

図 5.2　移動判定時の加速度データ

表 5.5　主作業時間の計測結果

製品	被験者名		第2回		第3回	
	第2回	第3回	真値 [s]	分析結果 [s]	真値 [s]	分析結果 [s]
クルマ（組立）	被験者2-1	被験者3-1-1	55.53	46.92	40.78	41.82
	被験者2-2	被験者3-2-1	46.61	41.63	32.47	36.91
	被験者2-3	被験者3-3-1	44.68	40.54	35.13	38.93
クルマ（分解）	被験者2-4	被験者3-2-2	32.23	33.60	35.20	37.51
	被験者2-5	被験者3-3-2	35.20	37.40	34.95	36.50
	被験者2-6	被験者3-1-1	32.66	33.96	40.20	43.64

5.3.4　主作業時間の計測と活用

第 2 回と第 3 回実験に最も良い結果となった条件 5 を適用したときの，各被験者の主作業時間の計測結果を表 5.5 に示す．表 5.5 の同じ行は，第 2 回と第 3 回実験の同一被検査の結果を示す.

結果より，従来のビデオカメラの映像分析による動作分析では計測が厳しい，同一範囲内で複数人が作業をする第 3 回実験の 2 人協力用セルにおいても，本システムでは個人を識別し主作業時間の計測ができていることが分かる．また，同製品を担当した被験者の主作業時間を比較すると，クルマ（組立）では第 2 回実験は被験者 2-3，被験者 2-2，被験者 2-1 の順，第 3 回実験は被験者 3-2-1，被験者 3-3-1，被験者 3-1-1 の順と，真値と分析結果で同様の結果が得られている．またクルマ（分解）でも，第 2 回実験は被験者 2-4，被験者 2-6，被験者 2-5 の順，第 3 回実験は被験者 3-3-2，被験者 3-2-2，被験者 3-1-1 の順と，真値と分析結果で同様の結果が得られている．さらに，同一被験者における第 2 回と第 3 回実験間での主作業時間を比較すると，被験者 2-4 と被験者 2-6 では増加し，それ以外の被験者では減少するという変化が，真値と分析結果の両方で確認できる．このように，加速度分析による主作業時間の計測は，映像目視確認による計測と同様の傾向の計測が可能な手法であることが分かる.

さらに，これらの結果を活用すると，以下のような現場改善や作業者教育の活動に繋がると考えられる.

1)　同一製品を担当する各作業者の主作業時間の比較が可能なことから，

- ムリのない作業速度の作業者の手順を参考にした新しい作業標準の作成・展開による，作業効率化および平準化による製造バラつき低減
- 作業者教育が必要なレベルか否かの判断に用いることによる，効果的な教育計画の作成

2) 時間経過による同一作業者の主作業時間の変化の確認が可能なことから，

- 訓練中の作業初心者の習熟度として扱い，実際の製造ラインの担当が可能か否かの判断による，製品品質低下や納期遅延の事前防止
- 作業者ごとの作業速度・スキルレベルを取り入れた製造スケジューリングへの適用による，スケジューリング誤差の低減
- 体調や疲労による作業者の主作業時間の変化を捉えることで，不注意による危険な状況の発生や非効率な働きすぎの抑止

被験者ごとの主作業時間の計測は，特に移動距離が短くて複数人が同時に作業することもある組立セル製造ラインにおいては，従来では熟練者による目視でしか行えない．本研究と同様の 12 名分のデータ収集の場合，熟練者も同じ時間の労力をかける必要があるため，定常的に行うことは困難である．本システムはセンサーを装着していれば複数作業者でも動作データをリアルタイムに収集・分析するため，従来よりも計測コストを大幅に低減させることが可能で，製造現場にとって有用なシステムであると考えられる．

5.4. まとめ

本章では，提案システムのデータ分析部で実行される処理の 1 つである，作業者動作データから作業者の移動と停止を推定することにより，作業履歴である製造中の異常動作の有無と，主作業時間の計測について述べた．

作業者に装着したウェアラブルセンサーの加速度の変化を分析することで動作を推定できることを利用して，作業者の移動と停止の状態を推定することで，作業履歴である製造中の異常動作の有無と主作業時間の計測を可能にした．分析はデータ前処理に無し・MAF・SGF の 3 種類，腰の加速度の成分分解のあり・無しの 2 種類を組み合わせた手法で行い，第 1 回実験の作業者動作データで条件探索をして，最も良い条件を第 2 回と第 3 回実験の作業者動作データへ展開した．

結果として，データ前処理に前後 20 個のデータで多項式への当てはめを計算する SGF 手法，腰の加速度の水平成分抽出を行ってから加速度の微分値である躍度を計算したうえで，腰の加速度の閾値 0.01 とし足首の加速度の閾値 0.12 で判定する手法の組み合わせが，作業履歴の計測精度を向上できることを明らかにした．一方で，加速度データのサンプリングレートを十分大きくすることが分析に重要なことを示した．さらに，移動と停止の状態推定から各被験者の主作業時間を計測した結果，真値である映像目視分析の結果と同様の傾向を示しており，計測結果を活用することで現場改善や作業者教育といった活動に繋がることを示した．

第 6 章　システムの応用例と展開性

本章では，本研究で構築したシステムのデータ分析部にて計測した製造進捗や作業履歴や主作業時間を用いた応用を示すとともに，本システムの限界や展開性について述べる．はじめに応用例の 1 つとして製造進捗や作業履歴をエージェント・シミュレータに適用して製造ノルマの完了時刻の予測を行った場合の効果について述べ，つぎに本研究で構築したシステムの限界について示し，最後に作業進捗や作業履歴や主作業時間を応用する手法や得られる効果と展開性について述べる．

6.1.　応用例：組立セル製造ラインのエージェント・シミュレーション

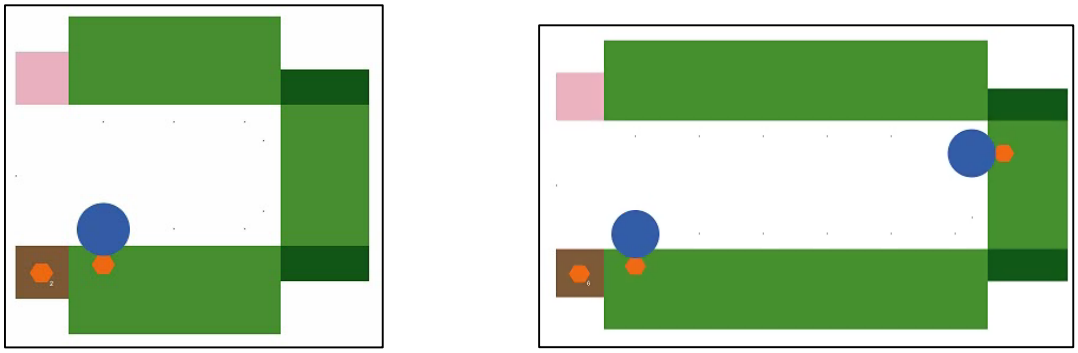
本システムで計測したデータの応用例として，エージェント・シミュレータを用いて，製造ラインにおける製造進捗や作業履歴から製造ノルマの完了時刻を予測し，ムダを発生させないための意思決定に繋げる例を示す．

6.1.1　エージェント・シミュレーションとは

エージェント・シミュレーション（Agent-based simulation）とは，「エージェント」と呼ばれる内部状態と意思決定機能を持つ動作主体が相互に作用しながら振る舞うことで，社会や経済といった複雑システムの性質をボトムアップに解析することに適した手法である[寺野 2004]，[寺野 2006]．

製造業は多数の工場が物流や情報面で連結したサプライ・チェーンと呼ばれる大規模で複雑システムを構成している．また，製造現場という範囲に限定しても，材料・設備・作業者という多くの要素が動的に絡み合いながら多種の製品を製造していく複雑なシステムとなっており，解析や予測にエージェント・シミュレーションが適している．これまでにもエージェント・シミュレーションを用いて，サプライ・チェーンのロバスト性評価，材料搬送の最適化，製造現場の改善評価，製造現場のエネルギーコスト最適化のための生産計画調整など，様々な工場における事例が報告されている[谷口 2011]，[菊池 2007]，[吉田 2014]，[則竹 2015]，[紺野 2016]．

本研究では，作業者をエージェントとして，組立セル製造ラインにおける製造ノルマの完了時刻を予測できるエージェント・シミュレーションを構築する．このシミュレータであ，提案システムにより作業者動作データを分析することで計測した製造進捗と作業履歴をインプットすると，作業者エージェントの移動速度や作業速度が標準作業の設定値から補正される．これにより，製造ノルマの完了時刻の予測値が，標準作業の設計値のみを用いた予測値よりも誤差が低減されることを狙っている．なお，本研究は 1 つの組立セル製造ラインだけを対象としており，さらに作業者エージェント間の相互作用が微小なため，エージェント・シミュレーションの有用性が低減している．しかしながら，設備と作業者



(a)　1 人用セル　　(b)　2 人協力用セル

図 6.1　エージェント・シミュレーションにおける製造フィールド

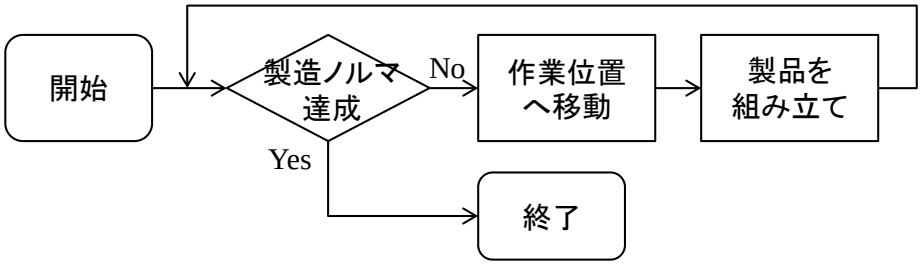


図 6.2　作業者エージェントの行動フロー

が協調しながら製造するような組立セル製造ラインの導入や，複数の組立セル製造ラインと倉庫の部品運搬者が同時に働いている製造フロア全体に，将来的に展開する可能性を考慮して，エージェント・シミュレーションによる実装を行った．

6.1.2　構築モデル

本シミュレーションの構成要素は，作業フィールドと作業者エージェントの 2 要素である．作業フィールドとしては，実際の組立セル製造ラインと同様のレイアウトとサイズを模して，作業机と作業時の立ち位置を内部情報として設定する（図 6.1）．作業者エージェントは事前に標準作業の手順と時間とライン内での作業位置をセットアップ情報として入力された上で，図 6.2 の処理フローに従って行動していく．2 人協力用セルにおけるセットアップ情報の例を表 6.1 に示す．なお，表 6.1 における PlaceID は各作業をどの作業机で行うかを示すものであり，作業フィールドとして作業机と立ち位置を設定する時につける ID と対応させるようにする．

作業者エージェントにおいて，作業位置への移動速度と製品を組み立てる作業速度には，はじめはセットアップ情報の標準作業時間が設定されるが，計測された作業者の実績デー

表 6.1　入力するセットアップ情報の例（2 人協力用セル）

WorkID	Worker	PlaceID	WorkTime	WorkDetail
1	1	101	1.14	ベース取り出し
2	1	101	9.84	フロントボディ
3	1	102	7.38	ミドルボディ
4	1	103	9.84	リアボディ
5	1	104	4.92	サイドミラー
6	1	105	4.92	ガラス・ウィング
7	1	106	5.16	バンパー
8	1	107	4.92	タイヤ
9	1	107	5.16	ボトム
10	1	107	0.78	完成品納入
1	2	108	1.14	完成品取り出し
2	2	108	8.88	ランプ・バンパー
3	2	109	4.32	タイヤ
4	2	109	5.64	ボトム
5	2	110	4.56	ガラス・ウィング
6	2	111	4.32	サイドミラー
7	2	112	9.00	リアボディ
8	2	113	6.66	ミドルボディ
9	2	114	9.00	フロントボディ
10	2	114	0.90	ベース納入

タがインプットされた時に移動速度と作業速度は入力データである実績に合うように補正される．ここで，i 個の製品が製造されて製造進捗であるサイクルタイム $\boldsymbol{C}=\{c_1, c_2, \ldots, c_i\}$[s] が得られたとする．このとき，$\boldsymbol{C}$ の平均値を Ac，$\boldsymbol{C}$ の標準偏差を Sc とおくと，k 番目に行う作業の速度の補正後平均値 Aa_k は

$$Aa_k = \frac{Ac}{\sum S_k} \times S_k \tag{6.1}$$

で表される．ここで，S_k は k 番目の作業の標準作業時間を表す．また，k 番目に行う作業の速度の補正後標準偏差 As_k は，

$$As_k = S_c \times S_k \times \sqrt{\frac{1}{\sum (S_k)^2}} \tag{6.2}$$

で表される．なお，As_k は S_k に比例するとの仮定を用いて計算している．

6.1.3　実験環境

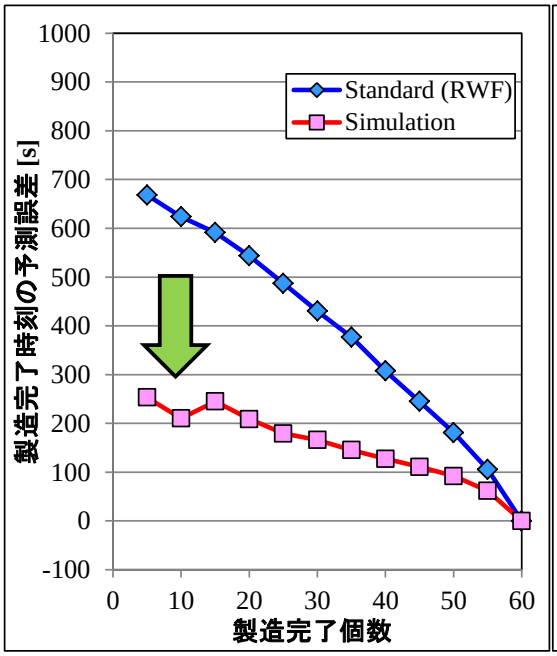
構築した組立セル製造ラインのエージェント・シミュレーションを利用して，標準作業時間を用いて製造ノルマの完了時刻を予測する手法と，本システムで計測した製造進捗を用いて製造ノルマの完了時刻を予測する手法を比較した．手法の評価は，各手法で予測した製造ノルマの完了時刻の予測値と実際の製造ノルマ完了時刻との誤差で行った．想定する環境としては 2 人協力用セルにおいて製造実験をした第 3 回実験を用いた．

各手法の予測は製品の製造が 5 個完了するたびに行うこととし，合計 60 個を製造ノルマとして完了時刻を予測した．例えば，製品の製造が 20 個完了した時の予測では，残り 40 個分の製造にかかる時間をそれぞれ予測する．なお，シミュレーションは各タイミングで 3 回ずつ計算し，3 回の平均値を予測結果として扱う．

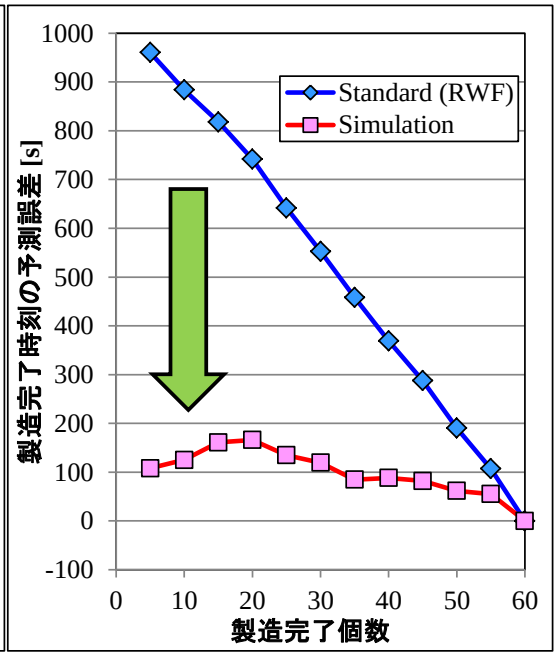
6.1.4　実験結果と考察

被験者ペアごとのシミュレーション結果を図 6.3 に示す．製造完了時刻の予測誤差とは，各手法の予測値と実際の製造ノルマ完了時刻の差であり，製造完了個数 60 個のときは残りの生産予定が 0 個のため両方の予測手法で誤差が 0 になる．結果より，いずれの被験者ペアにおいても，本システムで計測した製造進捗データとエージェント・シミュレーションを用いることで，製造開始後の早い段階から製造ノルマの完了時刻を少ない誤差で予測できていることが分かる．また，シミュレーションの所要時間は，Core (TM) i7-3540M CPU 3.0 GHz で RAM 4.0 GB のノートパソコンを用いて，1 回 1 秒程度であった．実験製品の標準作業時間は 64 秒程度であるため，リアルタイムで計算させるのに十分な余裕があることが確認できた．

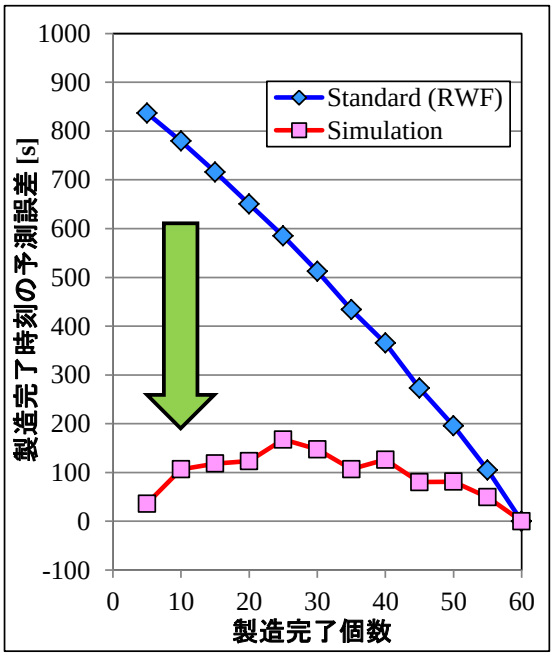
この結果を製造現場で用いる場合を考える．製造現場における製造完了時刻は基本的に標準作業時間による予測値でスケジューリングされている．したがって，本実験のように作業者の作業速度が標準作業時間よりも早い場合は，次に製造予定の製品部品準備や後工程への受入準備を前倒しさせる指示を出すといった意思決定に活用できると考えられる．一方で，本シミュレーションの範囲では発生していないが，作業者の作業速度が標準作業よりも遅い場合は，製造を早く終わらせるためのサポート人員投入や，遅れにより後工程において発生する待ち時間に行う軽作業の指示といった意思決定に活用できると考えられる．以上から，提案システムによって計測した製造進捗データとエージェント・シミュレーションを組み合わせる応用手法は，製造現場においても有用な効果を発揮できると期待できる．



(a) 被験者ペア 1 の結果



(b) 被験者ペア 2 の結果



(c) 被験者ペア 3 の結果

図 6.3　製造ノルマの完了時刻の予測結果

6.2. システムの限界

本研究で構築したシステムは，組立セル製造ラインを対象と定めたうえで，作業者動作データからの製造進捗と作業履歴や主作業時間の分析手法を検討し，計測を実現させた．ここでは本システムの限界について述べる．

本システムで分析している内容は下記の 3 項目である．

製造進捗　　：　定常的に移動する作業者がアンテナに近づくタイミングの抽出
作業履歴　　：　作業者の作業内容と作業位置が標準作業の指定どおりか否かの判定
主作業時間　：　作業者の停止時間の計測

これらの内容より，本システムが性能を発揮するために必要な条件を抽出すると，

1) 計測したい作業位置にアンテナを設置できる
2) 移動をする作業者が存在し，センサーを装着できる
3) 作業内容と作業位置を対応付けた標準作業が設定されている
4) 移動と作業を独立して同時に行う工程がない

という項目があげられる．

本システムの電波受信強度の分析による製造進捗は，製品の製造開始時に作業者が特定の位置に移動することを利用して導出している．したがって，本システムの計測対象とする作業者は，条件 2 に示すように移動をしている必要がある．実際の製造現場を考えると，ある特定の作業を連続で行うために移動がない作業者もおり，このような作業者には本システムを適用することができない．これは本システムに実装している分析手法の限界であるといえる．

また，本研究では組立セル製造ラインにおいて製品を直接製造する部分を実験したため，主作業と付随作業である移動のみの作業者動作データが計測対象となったが，実際の作業者の作業構成には付帯作業や余裕時間といった他の項目がある（図 6.4）．したがって，本システムを活用するためには，設備段取りなどの各付帯作業と作業位置の関係や，事務処理といった各余裕と作業者位置の関係といった事前準備が必要となる．

さらに，実際の製造現場においては，付帯作業である設備段取りを行ったあとにその設備を用いて主作業である製品加工を行うといった，作業者が移動せずに異なった作業を続けるような場面がある．本研究のシステムでは，設備付近にアンテナを設置することで，設備に近づいてから作業を終えて離れるまでの作業時間は計測することが可能だが，前述の付帯作業と主作業を切り分けて計測することができない．これは，本システムに実装している加速度データにより「移動」と「停止」を判定する分析手法の限界である．ただし，この問題は，加速度データの収集部位やサンプリングレートを改善して分析手法をより精緻にすることで十分解決が見込めるため，今後の課題であると考えられる．

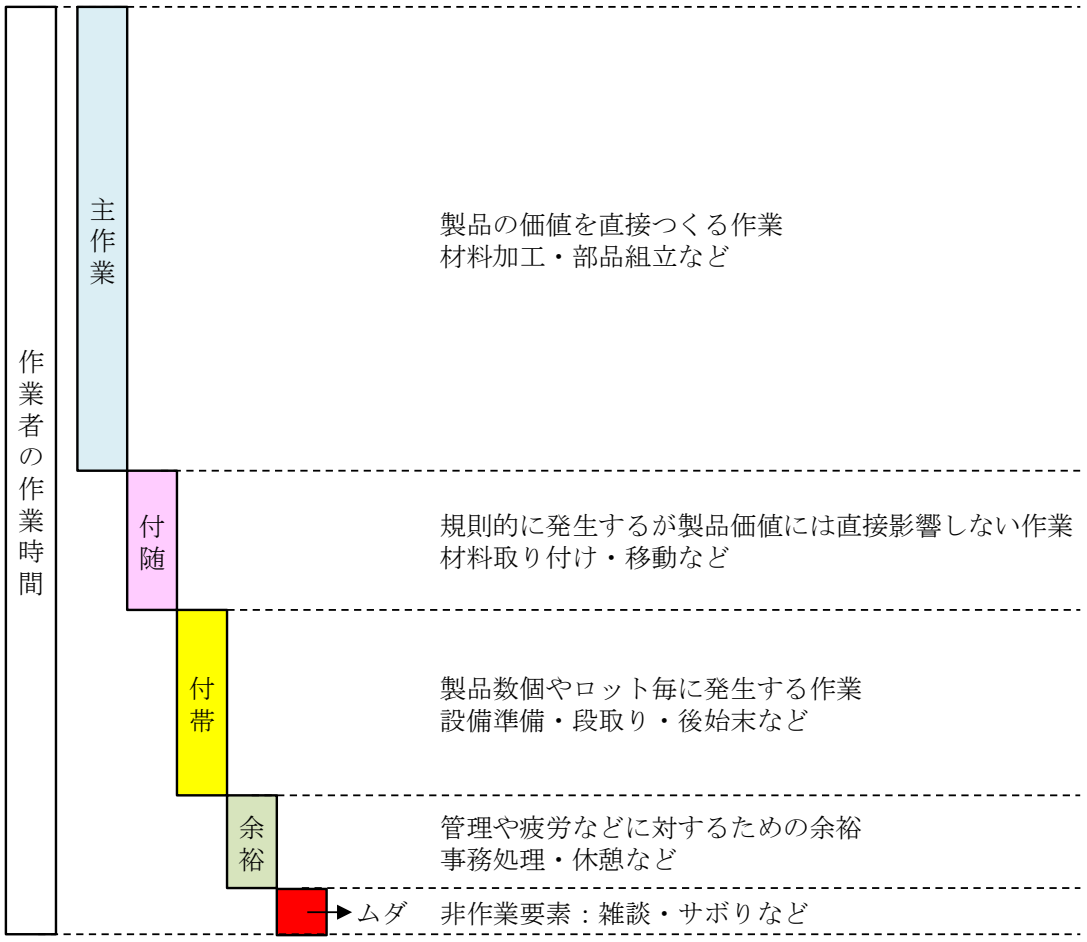


図 6.4　作業者の作業構成

6.3.　システムの応用と展開性

前節の限界を踏まえたうえで工場の各製造形態への展開性を考えると，表 6.1 のようにまとめられる．本研究でも対象としたセル生産やジョブショップ生産は基本的に各条件を満たすことが可能で本システムの展開が可能であると考えられる．しかし，ジョブショップ生産では設備を用いるため，設備稼働時に発生する電磁波による RSSI のノイズ量の変化確認や，設備の段取り作業後に直接加工を行うような同じ場所に留まって連続的に行う作業も詳細に分析することが必要な場合に加速度データの収集条件や分析手法の改善が必要など，設備を用いるさいの留意点が考えられる．また，ベルトコンベアを用いるライン・ロット生産は作業者が移動せずに作業をし続けるため，本システムによる分析手法では製品 1 個の切り替わるタイミングが抽出できず，適用することができない．

次に，これらの製造形態のラインが複数集まって構成される製造フロア全体で業務にあたる作業者，たとえば部品運搬や設備保全の担当者の計測が可能かを考えると表 6.2 のようにまとめられる．適用条件のうち，作業者移動と作業内容と位置関係に問題はないが，アンテナ設置と移動・作業の独立性には留意が必要になると考えられる．アンテナ設置につ

表 6.1　工場の各製造形態への展開性

	適用条件				総合評価
	アンテナ設置	作業者移動	作業内容と位置	移動・作業独立性	
ライン・ロット生産	○	×	○	○	×
セル生産	○	○	○	○	○
ジョブショップ生産	○※1	○	○※2	○	○

※1：設備稼働時に Beacon の電波受信強度にノイズが発生する可能性が考えられるため，事前にセンサー機器の稼働状況を確認する必要あり

※2：設備段取りをする付帯作業と製品加工をする主作業の切り分けが必要な場合は，加速度データの収集条件や分析手法を改善する必要あり

表 6.2　製造フロア全体での作業者への展開性

	適用条件				総合評価
	アンテナ設置	作業者移動	作業内容と位置	移動・作業独立性	
製造フロア全体	△※1	○	○	△※2	△

※1：製造フロアの広さからアンテナ設置コストの増大が想定されるため，事前のコスト検討や RSSI 分析手法の変更が必要な可能性あり

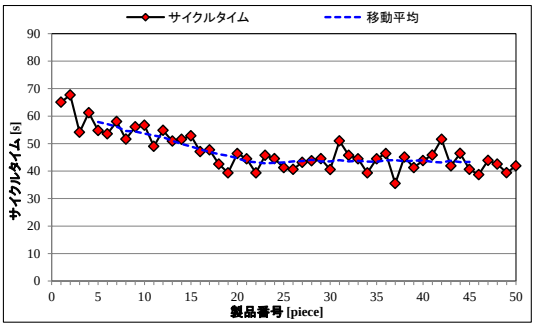
※2：軽い部品の取り出しや大きい設備の確認など移動しながら実施できる作業が想定されるため，事前確認が必要

いては，製造フロアは数十mから数百mの広さがあり通信可能距離が 10 m 程度の Beacon では大量のアンテナをフロアに配置する必要があるため，システム導入コストの点で難しい可能性があることに留意しなくてはならない．ただし，フロアが広く誤差が数 m あっても位置の推定が可能になるため，本システムのピーク抽出ではなく，RSSI 分析の従来研究のような複数アンテナとの距離から相対距離で位置を分析する手法を導入することで，解決できるとは考えられる．また，移動・作業の独立性については，倉庫での軽い部品の取り出しや大きい設備の目視確認など移動しながら遂行できる主作業が想定されるため，留意が必要である．しかしながら，これらの留意点の問題がなければ，適用条件は十分に満たすため，製造フロア全体での計測も問題なく達成できると考えられる．

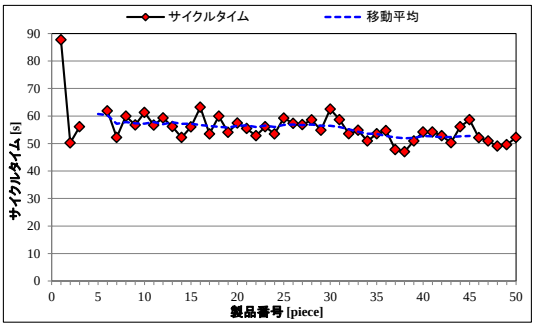
ここまで本システムの応用例と限界や展開をするさいの留意点について述べたが，一方で，本システムで得られる計測値を応用することで新たに可能になることも多く考えられる．本システムは，製造現場の製造ラインにおける作業者の製造進捗や作業履歴や主作業時間を逐次計測するものであるが，工場には他にも製造現場の材料や設備から収集できるデータや，経営戦略や調達と行った中長期なデータも存在する．ここでは，本システムで計測した項目の応用性，他の製造現場のデータと連携した場合の応用性，経営戦略などの中長期なデータと連携した場合の応用性について述べる．

本システムによりリアルタイムに計測できる作業者の製造進捗と作業履歴や主作業時間は，単独でも製造現場内の管理や効率の改善活動や教育に繋げることができる．まず，複数の製造ラインにおける製造進捗が逐次計測ができるため，図 6.5 のように製造ペースの変化を把握することが可能である．製品ごとやペア相手ごとなど各作業者の製造ペースの変動要因を明確にしたり，第 6.1 節の応用例で示したとおりに現在の製造ペースで納期遅れや手待ちが発生する可能性があるのかといった予測からムダを発生させないための意思決定に繋げたりすることができる．製造ペースの変化は，突発的な製造以外の業務により発生することもあるが，時系列による習熟や疲労によっても引き起こされる．このような習熟や疲労の影響は長時間に渡って計測することが必須であるため，従来の目視分析では実施が困難である．また，本システムは各作業者の製造進捗や主作業時間の値やバラつきを長期間に渡って容易に収集することができるため，作業者の育成に活用することが可能であると考えられる．たとえば，図 6.5(b)(c)(d)は同じクルマ組立を行った作業者の結果であるが，図 6.5(d)の作業者は他 2 名と比較して作業速度が遅くてバラつきも大きいことが分かるため，該当の作業者のみを育成するといったピンポイントな対応が容易になる．さらに，On the job training のような新たな製造スキルを身につけるさいの育成研修においても，本システムを用いて計測していれば研修の前後の結果を比較することができるため，研修が効果を及ぼしたかどうかの評価を容易に実現可能である．これは，育成研修だけでなく効率の改善活動の評価でも同様なことができ，本システムが製造現場の改善活動の推進に資することが可能なことを示していると考えられる．なお，本研究では作業履歴である製造中の異常の有無は加速度の分析により推定しただけであるが，異常を余計な作業の発生と考えると製造進捗であるサイクルタイムに影響を及ぼしている可能性がある．したがって，製造進捗と異常，特に品質異常との関係を分析することで，図 6.5 の各作業者の製造進捗のバラつきが問題のあるものかどうか判断が可能になることも期待できる．本研究で示してきた結果は合計 3 回の実験で収集したデータを用いた限定的な結果であるが，ここまでに示したとおり本システムで得られる計測値は単独でも多くのことに応用することが可能で，応用性が非常に高いシステムである．

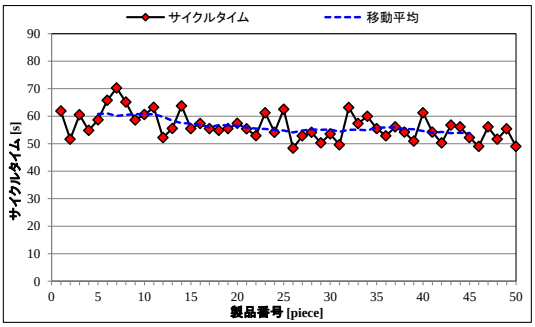
一方で，製造現場の他要素である材料や設備のデータと連携した場合の応用としては，製造スケジューリングや製品の原価や品質の管理に繋げることが考えられる．製造スケジューリングにおいては，従来の最適化アルゴリズムを用いた手法でも作業者の習熟度を仮



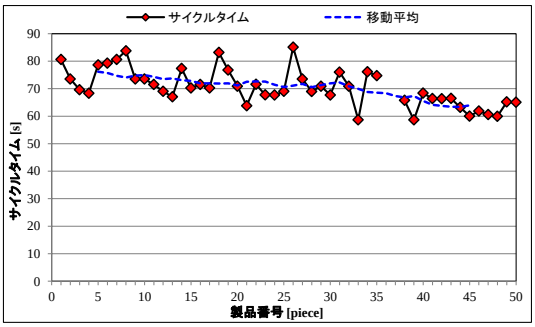
(a) クルマ分解作業者（図 4.8(b)と同じ）



(b) クルマ組立作業者その 1



(c) クルマ組立作業者その 2



(d) クルマ組立作業者その 3

図 6.5　本システムによる第 2 回実験における作業者 1 名ごとの時系列結果

※データがない部分は分析で製造進捗を計測できなかった部分

定値から実値へ置き換えることや，作業の所要時間を標準作業時間から作業者ごとの実績に合わせた値に置き換えることが可能で，これらはいずれも製造スケジューリングの予測誤差低減に効果的であると考えられる．また，製品の原価管理においても同様で，従来は仮想値を設定するしかなかった作業者に関する労務費のようなコスト指標が，実績に合わせた値に置き換えることが可能になり，コストダウンの必要性の検証や予算管理に効果的であると考えられる．

さらに，経営戦略などの中長期なデータと連携した場合の応用としては，セールス＆オペレーション・プランニングのような，経営と製造現場の情報を共有したうえで特定の指標で現在の状況を評価して，企業全体の迅速な意思決定に繋げるような手法との連携が考えられる．本システムにより普段から製造現場全体での製造能力を確認可能になるだけでなく，突発需要のような外乱発生時でも計測を続けることができる．そのため，経営戦略や様々な状況に対する対応を考えるにあたって，本システムで計測してきたデータを活用することで，実現可能性の高い経営戦略の立案や様々な外乱に対処できるロバストな工場の構築といった意思決定に非常に効果的であると考えられる．

6.4. まとめ

本章では，本研究で構築したシステムのデータ分析部にて計測した製造進捗や作業履歴や主作業時間を用いた応用と応用例について述べるとともに，本システムの限界や展開性について述べた．

本システムで計測した製造進捗の応用例として，作業者をエージェントとした組立セル製造ラインのエージェント・シミュレーションを構築し，製造ノルマの完了時刻の予測を行った．結果として，標準作業時間のみを用いて予測するよりも，計測した製造進捗を用いたエージェント・シミュレーションによる予測値のほうが，製造ノルマ完了時刻の予測誤差を低減できることが分かった．この応用手法は，製造現場にて次製品の部品準備前倒しや遅延している製造ラインの早期サポートといった意思決定への活用が期待できることを示した．

また，本研究で構築したシステムの限界としては，1) 作業者が作業と紐付いて移動をする必要がある，2) 移動を伴わない連続した作業の切り分けが必要な場合は加速度データの収集条件や分析手法の改善が必要である，という点を示した．

本研究で構築したシステムの展開性としては，セル生産やジョブショップ生産への展開は基本的に可能なものの，製造フロア全体では計測コストや移動と作業の独立性に留意が必要なことと，ベルトコンベアを用いるライン・ロット生産では作業者の移動がないため本システムが展開不可能なことを明確にした．また，計測項目の応用としては，計測した項目の応用性，他の製造現場のデータと連携した場合の応用性，経営戦略などの中長期なデータと連携した場合の応用性について述べた．計測した項目だけでも各作業者の作業変動要因の明確化や習熟や疲労の影響推定や作業者育成など幅広く応用が期待できるとともに，他データとの連携により改善活動や製造スケジューリングのように日常的に行われる活動への効果だけでなく，セールス＆オペレーション・プランニングのような経営戦略との連携にも効果が期待できることを示した．

第 7 章　結論と課題

本章では，本研究の結論と貢献，および，今後の課題について述べる．

7.1.　本研究の結論と貢献

本研究では「製造現場において，製造に関わる全ての要素の状態・動作データを，製品１個単位で，日常的に，もしくは，改善前後などの必要に応じて，容易に収集および分析をできる」という状態を構築するための，ウェアラブルセンサーを用いたリアルタイム作業者動作分析システムを提案した．

製造現場では納期や人的余裕の減少と製造に携わる熟練者の減少により，作業者の負担を軽減しつつ全体最適な改善を継続して実施できる環境の構築が求められている．しかしながら，作業者のセンシングにおいて従来の目視による計測手法では熟練者の専任が必要で計測コストが膨大になり定常的な計測が難しかった．

そこで本研究では，作業者動作のセンシングに Beacon と加速度センサーが組み合わさったウェアラブルセンサーを用いて，作業者の負担を抑えたうえで自動的に作業者の製造進捗と作業内容が得られるシステムを開発した．提案システムでは，作業者がウェアラブルセンサーを装着することで収集できる作業者動作データをリアルタイムに分析し，

a)　作業者による作業実績として製品 1 個単位で製造進捗が計測できる
b)　作業者による作業内容として作業履歴と主作業時間が計測できる
c)　複数人が同時に同じ場所にいても作業者ごとに個人特定できる
d)　長時間連続して計測できる

という機能を実装することで，作業者の負担を軽減しつつ，全体最適な改善を行うための製造現場の定常的な状況把握が可能な環境の構築を目指した．

本研究ではシステムが主に適用する製造形態を組立セル製造ラインに設定し，レゴブロックによる製品や U 字型の製造ラインを設計し，複数回の模擬製造実験で収集した作業者動作データを用いて各機能を実現する分析方法を導出した．

製品 1 個単位の製造進捗の計測は，データを収集するアンテナを組立セル製造ラインの作業開始位置に置くことで Beacon の RSSI が周期性を持つようになることを利用して, MAF によるデータ前処理とε-tube 法によるピーク抽出を組み合わせる分析手法で実現した．また理論値とノイズ量により想定される誤差を導出し，実現した分析手法は，周辺環境の影響によって生じる想定誤差の範囲内で計測できる手法であることを確認した．作業内容としての作業履歴と主作業時間の計測は，3 軸加速度センサーの値を用いて作業者の移動と停止の動作が推定できるようになることを利用して，SGF によるデータ前処理と腰の加速度の成分分解と躍度の閾値による判定を組み合わせる分析手法で実現した．得られた主作業時間は，実験時の映像を目視分析することで得た真値と同様の傾向が得られていることを確

認し，作業者育成や現場改善に繋がる計測値であることを示した．複数の作業者の個人特定と長時間連続した計測はウェアラブルセンサーの性能により実現しており，作業者の個人特定は通信内容として個別に設定できるセンサーID により実現可能にし，長時間の計測は省エネルギーな Bluetooth low energy beacon で通信するウェアラブルセンサーを用いること数か月から 1 年という稼働を実現した．これらの結果により，本研究にて提案したリアルタイム作業者動作分析システムは，作業者の負担を軽減しつつ，全体最適な改善を行うための製造現場の定常的な状況把握が可能な環境の構築に貢献する手法であることが示された．

さらに，本システムにて計測した製造進捗や作業履歴や主作業時間を用いた応用と応用例について述べるとともに，本システムの限界や展開性について述べた．応用例では本システムで計測した製造進捗を，作業者をエージェントとした組立セル製造ラインのエージェント・シミュレーションに適用して作業者の移動速度や作業速度に補正を加えることで，標準作業時間のみを用いて予測するよりも製造ノルマの完了時刻の予測誤差を低減できることを示し，本システムを用いることの有用性を示した．一方で，移動しない作業者の製造進捗を計測することができない点や，移動を伴わない連続した作業の切り分けが必要な場合は加速度データの収集条件や分析手法の改善が必要であるという点を，システムの限界として示した．また，本研究で構築したシステムの展開性としては，セル生産やジョブショップ生産への展開は基本的に可能なものの，製造フロア全体では計測コストや移動と作業の独立性に留意が必要なことと，ベルトコンベアを用いるライン・ロット生産では作業者の移動がないため本システムが展開不可能なことを明確にした．そして，計測項目の応用としては，計測した項目の応用性，他の製造現場のデータと連携した場合の応用性，経営戦略などの中長期なデータと連携した場合の応用性について述べた．計測した項目だけでも各作業者の作業変動要因の明確化や習熟や疲労の影響推定や作業者育成など幅広く応用が期待できるとともに，他データとの連携により改善活動や製造スケジューリングのように日常的に行われる活動への効果だけでなく，セールス＆オペレーション・プランニングのような経営戦略との連携にも効果が期待できることを示した．

7.2. 今後の課題

本システムの今後の課題として，以下が考えられる．

1) Beacon の RSSI 分析に三角測量の手法の拡張
2) 加速度分析の拡張
3) データ応用部の拡張
4) 実工場での活用による改善や効率化の実践

課題 1)について，本研究で構築したシステムはセル生産やジョブショップ生産への適用は可能であるが，製造フロア全体への適用時には複数アンテナとの距離を求めて三角測量を行う手法が効果的な可能性があると考察した．分析手法の 1 つとしてデータ分析部に実

装し，環境に合わせて分析手法を選択可能なシステムにすることで，製造現場でさらに活用しやすいシステムになると考えられる．

課題 2)については 2 種類が考えられる．1 つは，移動を伴わない連続した作業の切り分けが必要な場合にも対応できるようにするため，加速度データの収集部位やサンプリングレートを改善するとともに，より精緻な分析手法により複雑な動作の推定も可能にすることである．もう 1 つは，加速度データによる分析を動作推定だけでなく作業者の状況推定といった他項目の推定に用いることである．たとえば，移動中の動作の変化から疲労を推定することで，働きすぎの防止や疲労によって生じる作業速度や安全性低下を未然に防ぐ意思決定に繋がると考えられる．

課題 3)について，本研究ではデータ応用部の例として組立セル製造ラインのエージェント・シミュレーションを用いた製造ノルマ完了時間の予測誤差の変化を示したが，本システムで計測したデータは様々な活動に用いることができる．たとえば，エージェント・シミュレーションを複数ラインと部品作業者がいる製造フロア全体への拡張や，製造スケジューリングのアルゴリズムを導入や，セールス＆オペレーション・プランニングと連携するための経営指標への変換といった機能を拡張することで，より活用の幅が広がると考えられる．

課題 4)について, 本研究では実験により計測した製造進捗や作業履歴や主作業時間から，製造現場の効率化や改善に繋がることを示したが実践はできていない．これまでに示したとおり，本システムの応用の幅は非常に広い．本システムを実工場へ適用して導出された改善や効率化を実践することで，本システムの有用性のさらなる明確化に繋がるだけでなく，工場経営に資することができると考えられる．

参考文献

第 1 章

[Hopp2000] Hopp, W., and Spearman, M.: "Factory Physics," McGraw Hill Higher Education, 2nd Revised (2000)

[FMER2013] Final report of the Industrie 4.0 Working Group: "Recommendations for implementing the strategic initiative INDUSTRIE 4.0," Federal Ministry of Education and Research, Germany (2013)

[Evans2012] Evans, P.C. and Annunziata, M.: "Industrial Internet: Pushing the Boundaries of Minds and Machines," General Electric (2012)

[貝原 2016] 貝原俊也：IoT 環境下の「考える工場」実現を目指す実仮想融合型生産システム，計測と制御，Vol. 55，No. 1，pp. 53-58（2016）

[出口 2016] 出口弘：組織・産業・経済システムの人工物としてのデザイン論－IoE 時代の組織・産業・経済システムの現実の再構築に向けて－，計測と制御，Vol. 55，No. 1，pp. 59-70（2016）

[ANSI2005] ANSI/ISA-05.00.03-2005: "Enterprise - Control System Integration-, Part 3: Activity Models of Manufacturing Operations Management," International Society of Automation (2005)

[APSOM2015] ものづくり APS 推進機構（APSOM）: 製造オペレーションマネジメント入門～ISA-05 が製造業を変える，ものづくり APS 推進機構（2015）.
http://apsom.org/docs/2015/20150323poms.pdf（2017/06/10 最終確認）

[藤田 1997] 藤田彰久：新版 IE の基礎，建帛社（1997）

[泉 2014] 泉洋介：作業時間から工程・時限・反復効果を分離する統計モデルの研究－製造ラインの改善に向けて－，総合研究大学院大学博士学位論文（2014）

[久保 2007] 久保幹雄：サプライ・チェイン最適化ハンドブック，朝倉書店（2007）

[Taylor1911] Taylor, F.W.: "The Principles of Scientific Management," Harper & Brothers (1911)

[Gilbreth1911] Gilbreth, F.: "Motion Study," D. Van Nostrand Company (1911)

[Pigage1954] Pigage, L.C., Tucker, J.L.: "Motion and Time Study," Urnama. IL; University of Illinois (1954)

[川瀬 2007] 川瀬武志：IE 問題の基礎，日刊工業新聞社（2007）

[三次 2008] 三次仁，鈴木茂哉，羽田久一，稲葉達也：ネットワーク型 RFID の最新技術と利用動向，通信ソサイエティマガジン，電子情報通信学会，Vol. 7，No. 冬，pp. 30-40（2008）

[DSRI2010] 流通システム開発センター : RFID システム導入ガイドライン，財団法人流通システム開発センター（The Distribution Systems Research Institute）（2010）

[Altaf2015] Altaf, M. S., Yu, H., Liu, H., and Al-Hussein, M.: "Online Simulation Modeling of Prefablicated Wall Panel Production using RFID System," Proceedings of the 2015 Winter Simulation Conference, pp. 3379–3390 (2015)

[Sendon2015] Sendon, A., Pinaton, J. and Dauzere-Peres, S.: "Simulation Model to Control Risk Levels on Process Equipment through Metrology in Semiconductor Manufacturing," Proceedings the 2015 Winter Simulation Conference, pp. 2941-2952 (2015)

[鄭 2016] 鄭立：無線通信技術 BLE と製造業 IoT/M2M への適用，計測と制御，Vol. 55，No. 12，pp. 1036-1041（2016）

[竹林 2017] 竹林康太，浦口智貴，Chang Shuang，出口弘：生産管理のための疎結合アーキテクチャの研究，第 12 回社会システム部会研究会，計測自動制御学会，pp. 174-184（2017）

[Groover2013] Groover, M. P.: "Work Systems: Pearson New International Edition: The Methods, Measurement and Management of Work," Pearson (2013)

[李沢 2012] 李沢亜土里，大和田勇人：複数の監視カメラからの画像を用いた工場内での人の動き分析に関する研究，情報処理学会第 74 回全国大会，Vol. 6S-9，pp. 2-431-2-432（2012）

[井坂 2016] 井坂英也，永吉洋登，吉川裕，山田敏広，掛布修弘：センシング技術と解析技術を活用した次世代グローバル製造管理，日立評論，Vol. 98，No. 3，pp. 184-185（2016）

[小暮 2008] 小暮潔：看護業務のセンシング，人工知能学会誌，Vol. 23，No. 4，pp.468-473（2008）

[Thiesse2009] Thiesse, F., Al-Kassab, J., and Fleisch, E.: "Understanding the value of integrated RFID systems: a case study from apparel retail", European Journal of Information Systems, pp.1-23 (2009)

[新村 2012] 新村猛，竹中毅，上岡玲子，蔵田武志，大浦修一：外食産業におけるサービス工学の導入事例，精密工学会誌，Vol. 78，No. 3，pp. 208-211（2012）

[Boyd2011] Boyd, L.J., Ball, K., and Aughey, R.J.: "The Reliability of MinimaxX Accelerometers for Measuring Physical Activity in Australian Football," International Journal of Sports Physiology and Performance, Vol. 6, Issue 3, pp. 311-321 (2011)

[Nagano2014] Nagano, H., James, L., Sparrow, W.A., and Begg, R.K.: "Effects of Walking-induced Fatigue on Gait Function and Tripping Risks in Older Adults," Journal of NeuroEngineering and Rehabilitation, Vol. 11, No. 155, pp. 1-7 (2014)

[谷水 2003] 谷水義隆，坂口龍彦，杉村延広：遺伝的アルゴリズムを用いたリアクティブスケジューリング（第 1 報，作業の遅延に対する生産スケジュールの変更），日本機械学会論文集（C 編），Vol. 69，No. 685，pp.234-239（2003）

[八尾 2011] 八尾佳宏，貝原俊也，藤井信忠，堀栄一：動的生産環境下におけるスケジューリングの自動化に関する研究，日本機械学会論文集（C 編），Vol. 77，No. 784，pp.395-406（2011）

[高橋 2015] 高橋啓太，小野里雅彦，田中文基：動的変化に適応する生産システムの統合的計画手法に関する研究（第 3 報）：ZDD の連結による解空間表現とその解法の検討，日本機械学会生産システム部門講演会講演論文集，Vol. 2015，pp. 85-86（2015）

[舘野 2007] 舘野寿丈，清水慶子：作業者の技能レベルと教育を考慮した作業スケジューリング支援，日本機械学会論文集（C 編），Vol. 73，No. 734，pp.2854-2862（2007）

[岩村 2014] 岩村幸治，西濱大佑，田中健太郎，谷水義隆，杉村延広：生産計画を考慮した作業者の教育計画の作成手法に関する研究，日本機械学会論文集，Vol. 80，No. 814，pp. 1-12（2014）

第 2 章

[Billo1998] Billo, R.E.: "A Design Methodology for Configuration of Manufacturing Cells," Computers & Industrial Engineering, Vol. 34, Issue 1, pp. 63-75 (1998)

[Johansen1995] Johansen, J., and Mikkeisen, H.: "The Lego Truck Game - A game of production control," J. O. Riis(ed.), Simulation Games and Learning in Production Management, Springer US, pp. 127-133 (1995)

[Dong2014] Dong, Y., Haraguchi, H., and Hao, X.: "Structural Equation Modelling of Human Factors and Their Impact On Productivity of Cellular Manufacturing," Innovation and Supply Chain Management, Vol. 8, No. 1, pp. 1-7 (2014)

[Tanimizu2015] Tanimizu, Y., Ishii, S., and Yokotani, T.: "A study on development of a work instruction system for assembly cells based on analysis of learning processes," Journal of Advanced Mechanical Design, Systems, and Manufacturing, Vol. 8, No. 4, pp. 1-10 (2014)

[Tanner1990] Tanner, J.P., "Manufacturing Engineering: AN INTRODUCTION TO THE BASIC FUNCTIONS, SECOND EDITION, REVISED AND EXPANDED," CRC Press (1990)

[吉田 2016] 吉田良耿：RWF 法（Ready Work-Factor）とは，（一財）大阪科学技術センター技術振興部 ATAC 事務局．http://www.atac.ne.jp/others/rwf_1.pdf（2017/06/10 最終確認）

[Chalder1993] Chalder, T., Berelowitz, G., Pawlikowska, T., Watts, L., Wessely, S., Wright, D., and Wallace, E.P.: "Development of a Fatigue Scale," Journal of Psychosomatic Research, Vol. 37, No. 2, pp. 147-153（1993)

[酒井 2002] 酒井一博:日本産業衛生学会産業疲労研究会撰「自覚症しらべ」の改訂作業 2002」, 労働の科学, Vol. 57, No. 5, pp. 295-298（2002）. http://square.umin.ac.jp/of/jikakusyou-sirabe2.pdf（2017/06/10 最終確認）

[産業疲労研究会 2016] 産業疲労研究会：作業改善のための調査ツール「疲労部位しらべ」, 日本産業衛生学会産業疲労研究会．http://square.umin.ac.jp/of/hiroubui.pdf（2017/06/10 最終確認）

[Scott1976] Scott, J., and Huskisson, E.C.: "Graphic Representation of Pain," Pain, Vol. 2, Issue 2, pp. 175-184 (1976)

第 4 章

[Apple2014] Apple Developer: "Getting Started with iBeacon," iBeacon - Apple Developer, pp. 1-11 (2014). https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf（2017/06/10 最終確認）

[佐藤 2011] 佐藤智美，小宮山哲，下田雅彦，劉渤江，横田一正：Bluetooth の電波強度を用いた位置推定方式の検討，第 3 回データ工学と情報マネジメントに関するフォーラム，Vol. B9，No. 4，pp. 1-6（2011）

[川村 2011] 川村諒，久保田真一郎，福島慶人，古川誠一，杉谷賢一：既設アクセスポイントを利用した屋内位置情報取得システムのための位置推定精度による分析，情報処理学会論文誌，Vol. 52，No. 3，pp. 1357-1364（2011）

[春本 2011] 春本要，藤原謙太郎，寺西裕一，秋山豊和，竹内亨，西尾章治郎：存在確率分布の伝播を用いた自己位置推定手法，情報処理学会論文誌，Vol. 52，No. 5，pp. 1862-1870（2011）

[Bahl2000] Bahl, P., and Padmanabhan, V.N.: "RADAR: an in-building RF-based user location and tracking system," 19th Annual Joint Conference of the IEEE Computer and Communications Societies, Tel Aviv, Israel, pp. 775-784 (2000)

[Chen2012] Chen, Y., Lymberopoulos, D., Liu, J., and Priyantha, B.: "FM-based Indoor Localization," 10th international conference on Mobile systems, applications, and services, Lake District, UK, pp. 169-182 (2012)

[坂 2014] 坂涼司，梶克彦，河口信夫 : 磁気と WiFi 電波強度を含んだマップ情報に歩行者デッドレコニングを併用した屋内位置推定手法，信学技法，ASN2013-122，23/28（2014）

[北澤 2010] 北澤正樹，高橋雅和，山田隆志，吉川厚，寺野隆雄 : RFID 技術による小売店販売シミュレーターの顧客回遊行動の分析，2010 年度第 24 回人工知能学会全国大会，人工知能学会，（2010）

[藤野 2013] 藤野俊樹，北澤正樹，山田隆志，高橋雅和，山本学，吉川厚，寺野隆雄：エージェントシミュレーションと実データに基づく小売店舗内における顧客行動に関する研究，第 4 回社会システム部会研究会，計測自動制御学会（2013）

[大畠 2013] 大畠一人，金沢孝：動線分析機器を用いた設備運転における作業パターンの分析手法に関する研究，ヒューマンファクターズ，Vol. 18，No. 1，pp.27-41（2013）

[Orfanidis2010] Orfanidis, S. J.: "Introduction to Signal Processing," Rutgers University (2010). http://www.ece.rutgers.edu/~orfanidi/intro2sp/orfanidis-i2sp.pdf（2017/06/10 最終確認）

[村尾 2010] 村尾和哉，クリストフラールホーフェン，寺田努，西尾章治郎：センサのピーク値を用いた状況認識手法，情報処理学会論文誌，Vol. 51，No. 3，pp. 1068-1077（2010）

[村尾 2011] 村尾和哉，寺田努：加速度センサの定常性判定による動作認識手法，情報処理学会論文誌，Vol. 52，No. 6，pp. 1968-1979（2011）

第 5 章

[中川 2016] 中川勝哉，高橋 B.徹，高橋聡，宮部博史：避難訓練における個人の行動特性分析のセンサデータを使った簡易化の提案，第４回ビジネス・インフォマティクス研究会，人工知能学会，pp. 1-5（2016）

[Najafi2003] Najafi, B., Aminian, K., Paraschiv-Ionescu, A., Loew, F., Bula, C. J. and Robert, P.: "Ambulatory System for Human Motion Analysis Using a Kinematic Sensor: Monitoring of Daily Physical Activity in the Elderly," IEEE Transactions on Biomedical Engineering, Vol. 50, No. 6, pp. 711-723 (2003)

[小林 2006] 小林哲平，三宅美博，和田義明，松原正明：加速度センサを用いた運動学的歩行分析システムー股関節疾患の術後リハビリにおける Walk-Mate 有効性評価への適用ー, 計測自動制御学会論文集，Vol. 42，No. 5，pp. 567-576（2006）

[Gafurov2009] Gafurov, D., and Snekkenes, E.: "Gait Recognition Using Wearable Motion Recording Sensors," EURASIP Journal of Advances in Signal Processing, Vol. 2009, pp. 1-16 (2009)

[黄 2017] 黄冬陽，相田晋，北澤正樹，高橋聡，高橋 B.徹，吉川厚，寺野隆雄：ウェアラブルセンサーを用いた機械学習による避難評価手法，第 12 回社会システム部会研究会，計測自動制御学会，pp. 52-61（2017）

[Wang2005] Wang, S., Yang, J., Chen, N., Chen, X. and Zhang, Q.: "Human Activity Recognition

with User-Free Accelerometers in the Sensor Networks," International Conference on Neural Networks and Brain, Vol. 2, pp. 1212-1217 (2005)

[Koskimaki2009] Koskimaki, H., Huikari, V., Siirtola, P., Laurinen, P., and Roning, J.: "Activity Recognition using a Wrist-worn Inertial Measurement Unit: A Case Study for Industrial Assembly Lines," 17th Mediterranean Conference on Control and Automation, Thessaloniki, Greece, pp. 401-405 (2009)

[Altun2010] Altun, K., and Barshan, B.: "Human Activity Recognition using Inertial/Magnetic Sensor Units," International Workshop on Human Behavior Understanding, Springer, pp. 38-51 (2010)

[Joshua2011] Joshua, L., and Varghese, K.: "Accelerometer-Based Activity Recognition in Construction," Journal of Computing in Civil Engineering, Vol. 25, Issue 5, pp. 370-379 (2011)

[Tontani2013] Tontani, Y., Kajiwara, Y., Harada, F. and Shimakawa, H.: "Judging Working Rhythm from Body Movement to Prevent Human Errors," Recent Researches in Telecommunications, Informatics, Electronics and Signal Processing, WSEAS LLC, pp. 127-132 (2013)

[矢野 2014] 矢野和男：データの見えざる手，草思社（2014）

[村上 2014] 村上知子，鳥居健太郎，長健太，内平直志：センサデータと業務知識からのトピックモデルを用いた看護業務行動の推定, 人工知能学会論文誌, Vol. 29, No. 5, pp. 427-435（2014）

第 6 章

[寺野 2004] 寺野隆雄：エージェント・ベース・モデリングへの招待，オペレーションズ・リサーチ：経営の化学，Vol. 49，No. 3，pp. 131-136（2004）

[寺野 2006] 寺野隆雄：エージェント・ベース・モデリングの技術と応用，社会シミュレーションの技術敵動向と実務的課題，計測自動制御学会システム工学部会・経営情報学会合同シンポジウム資料，pp. 1-8（2006）

[谷口 2001] 谷口憲, 倉橋節也, 寺野隆雄 : エージェントに基づくサプライチェーンモデル, 情報処理学会研究報告（知能と複雑系研究会），Vol. 2001，No. 1，pp. 109-114（2001）

[菊池 2007] 菊池純二，小西正躬，今井純：自律分散エージェントを用いた原料ヤード内鉱石の動的搬送経路計画，鉄と鋼，Vol. 90，No. 10，pp. 11-19（2007）

[吉田 2014] 吉田琢史，林久志，西成活裕：シミュレーションによる最適化手法を用いた大規模ジョブショップ型工場における搬送効率の改善，コンピュータソフトウェア，Vol. 31，No. 3，pp.130-140（2014）

[則竹 2015] 則竹茂年：生産ラインシミュレーションを用いた Industrie4.0 の効果検証，日本機械学会生産システム部門研究発表講演会（2015）.
https://www.jsme.or.jp/msd/event/conference2015doc/P009_noritake.pdf（2017/06/10 最終確認）

[紺野 2016] 紺野剛史，梅宮茂良，吉田宏章，竹林知善，Shuang Chang，出口弘：多層基盤製造工場におけるエネルギーコスト最適化のための生産計画調整シミュレーション，第 10 回社会システム部会研究会，計測自動制御学会，pp. 93-98 （2016）

謝辞

本研究の作成にあたり，多くの方々に御支援を受けました．この場を借りて厚く御礼を申し上げます．

東京工業大学大学院の寺野隆雄先生には，博士課程の指導教官として研究の進め方や論文執筆だけでなく，他の企業とのテーマへの取り組みなど様々な経験をさせていただきました．また，吉川厚先生には，修士課程の指導教官としての研究の組み立て方や論文執筆だけでなく，幅広い分野の知識をつけることの楽しさを教えていただきました．研究を楽しみながら進めてこられたのも御二人の御指導のおかげです．さらに，東京工業大学の出口弘先生，新田克己先生，三宅美博先生，小野功先生，山村雅幸先生には論文審査を通して博士論文を執筆する上での貴重な御指摘，御助言をいただきました．皆様に深く感謝いたします．

共同研究者の東京理科大学の高橋徹先生と高橋聡先生には，研究テーマの構成や進め方から実験環境構築や学会発表資料の構成まで，様々なところで御助言をいただきました．構想としてずっと温めていただけの本研究テーマを実現すると決めてから 2 年強でここまで進められたのも，頻繁に情報共有や議論をさせていただいたおかげです．深く感謝いたします．同じ研究室に通っていた時代よりも，所属がわかれた今の方が深い付き合いになっていることを不思議に感じたりもしますが，これからも面白いことを一緒に進めさせてください．

山口大学の高橋雅和先生には，修士時代より研究構成やデータ分析，学会投稿など多くのところで御指摘や御助言をいただきました．先生の考え方や進め方は本研究を進める上でも大いに参考にさせていただいております．また，会社員時代の福田様，吉本様，井上様，宗近様，江上様，吉村様には，仕事を通じて製造現場における生の知識や課題を感じる機会や御助言をいただけました．厚く御礼申し上げます．

寺野研究室の方々にも様々な場面で御支援いただきました．秘書の山口様には実験材料準備や学会発表手配など多くの場面でお世話になりました．國上様をはじめとした先輩の皆様には多くの御指導をいただきました．後輩の皆様には実験協力をはじめ，分析でもお手伝いいただきました．皆様に深く感謝いたします．

博士号は研究を進めるにあたっての最低限を自力で行えるようになったというスタートラインを示すものだと考えています．これからも自分のスキルをたゆまず向上させるとともに，社会へ広く還元できる人間になるよう，ますます精進していきます．

2017 年 9 月末日

業績一覧

本論文に直接関わる業績

学術誌

Masaki Kitazawa, Satoshi Takahashi, Toru B. Takahashi, Atsushi Yoshikawa, and Takao Terano: "Real Time Workers' Behavior Analyzing System for Productivity Measurement using Wearable Sensor," Journal of Control, Measurement, and System Integration, Society of Instrument and Control Engineers (Nov. 2017).

査読付き国際会議

Masaki Kitazawa, Satoshi Takahashi, Toru B. Takahashi, Atsushi Yoshikawa, and Takao Terano: "Improving a Cellular Manufacturing System through Real Time-Simulation and -Measurement," SSERV2016 on COMPSAC2016, Atlanta, USA (Jun. 2016).

Masaki Kitazawa, Satoshi Takahashi, Toru B. Takahashi, Atsushi Yoshikawa, and Takao Terano: "Combining Workers' Behavior Data and Real Time Simulator for a Cellular Manufacturing System," IFMIP2016 on WAC2016, Rio Grande, Puerto Rico (Satellite: Himeji, Japan) (Jul. 2016).

国内学会・ポスター・研究会

Masaki Kitazawa, Satoshi Takahashi, Toru B. Takahashi, Atsushi Yoshikawa, and Takao Terano: "Real Time Simulator with Multiple Worker's Behavior Data for a Cellular Manufacturing System," Korea-Japan Social Simulation Workshop 2016 (JaKoSS 2016), Seoul, Korea, pp. 337-344 (Oct. 2016).

北澤正樹，高橋聡，高橋 B.徹，吉川厚，寺野隆雄：ウェアラブルセンサーを用いた作業者動作分析によるセル生産ラインのリアルタイム生産進捗計測システムの提案，第 12 回社会システム部会研究会，計測自動制御学会，pp. 337-344（Mar. 2017）（優秀ポスター賞）．

北澤正樹，高橋聡，高橋 B.徹，吉川厚，寺野隆雄：ビーコンデバイスと加速度センサによる製造現場作業者の動作計測とシミュレーション，2017 年度人工知能学会全国大会，人工知能学会（May 2017）．

競争的資金

高橋 B.徹，高橋聡，北澤正樹：避難訓練におけるセンサデータからの個人行動抽出アルゴリズムの開発，問題複合体を対象とするデジタルアース共同利用・共同研究，中部大学 中部高等学術研究所 国際 GIS センター，IDEAS201613（2016）.

博士後期課程中の業績

学術誌

Toshiki Fujino, Masaki Kitazawa, Takashi Yamada, Masakazu Takahashi, Gaku Yamamoto, Atsushi Yoshikawa, and Takao Terano: "Analyzing In-store Shopping Paths from Indirect Observation with RFID Tags Communication Data," RISUS - Journal on Innovation and Sustainability, Arnoldo Jose EDS. Vol. 5, No. 1, pp. 89-96 (Feb. 2014).

Masaki Kitazawa, Takashi Yamada, Masakazu Takahashi, and Takao Terano: "Analyzing Supermarket Shopping Factors from Indirect Observation and Simulation Study," RANGSIT JOURNAL OF INFORMATION TECHNOLOGY (Post-proceedings of ACIS2013), Rangsit University, Vol. 1, No. 2, pp. 1-5 (Jun. 2014).

査読付き国際会議

Masaki Kitazawa, Takashi Yamada, Masakazu Takahashi, and Takao Terano: "Agent-Based In-Store Simulator for Customer Pedestrian Behaviors in a Supermarket," The 8th International Workshop on Agent-based Approach in Economic and Social Complex Systems (AESCS2013), Tokyo, Japan (Sep. 2013).

Masaki Kitazawa, Fumiaki Sato, Takashi Yamada, Masakazu Takahashi, and Takao Terano: "How Do Customers Buy Them at a Supermarket? Behavior Analysis from Real Observation and Agent Simulation," The 2nd Asian Conference on Information Systems (ACIS2013), Phuket, Thailand (Oct. 2013).

Toshiki Fujino, Masaki Kitazawa, Takashi Yamada, Masakazu Takahashi, Gaku Yamamoto, Atsushi Yoshikawa, and Takao Terano: "Analyzing In-store Shopping Paths from Indirect Observation with RFID Tags Communication Data," International Conference on Innovation and Management 10th, PUC SP, Brazil (Dec. 2013).

Masaki Kitazawa, Masakazu Takahashi, Takashi Yamada, and Takao Terano: "Analyzing Supermarket Shopping Paths from Indirect Observation and Simulation Study," 2013 International Conference on Signal-Image Technology & Internet-Based Systems (SITIS2013), Kyoto, Japan, pp.939-943 (Dec. 2013).

Masaki Kitazawa, Tomohiro Yoshimura, Shin Egami, Manabu Kano, and Yoshitake Ito: "Yield Enhancement in MEMS Wafer Level Fabrication by Whole Process PLS Model," International Symposium on Semiconductor Manufacturing 2014 (ISSM2014), Tokyo, Japan (Dec. 2014) （共著者代理発表）.

Tomohiro Yoshimura, Masaki Kitazawa, Shin Egami, and Tomofumi Nakamura: "Predictive Control by Whole Process PLS Model in MEMS Fabrication," AEC/APC Symposium Asia 2015, Tokyo, Japan (Nov. 2015).

国内学会・ポスター・研究会

藤野俊樹，北澤正樹，山田隆志，高橋雅和，山本学，吉川厚，寺野隆雄：エージェントシミュレーションと実データに基づく小売店舗における顧客行動の分析，2013 年度人工知能学会全国大会，人工知能学会，Vol. 27，pp. 1-2（Jun. 2013）.

藤野俊樹，北澤正樹，高橋雅和，山田隆志，山本学，吉川厚，寺野隆雄：小売店舗内における顧客行動シミュレーションに関する研究，第 3 回社会システム部会研究会，計測自動制御学会，Vol. 3，pp. 125-128（Feb. 2013）.

藤野俊樹，北澤正樹，山田隆志，高橋雅和，山本学，吉川厚，寺野隆雄：エージェントシミュレーションと実データに基づく小売店舗内における顧客行動に関する研究，第 4 回社会システム部会研究会，計測自動制御学会（Sep. 2013）.

藤野俊樹，北澤正樹，山田隆志，高橋雅和，山本学，吉川厚，寺野隆雄：実データに基づく小売店舗内における顧客行動の分析，システム・情報部門 学術講演会 2013，計測自動制御学会，pp. 438-442（Nov. 2013）.

藤野俊樹，北澤正樹，山田隆志，高橋雅和，山本学，吉川厚，寺野隆雄：スーパーマーケットで客はどう動く? -顧客動線分析とエージェントシミュレーションからわかること-，第 5 回社会システム部会研究会，計測自動制御学会（Mar. 2014）.

北澤正樹，高橋雅和，寺野隆雄：高齢化による移動行動変化を考慮した都市動態エージェントシミュレーション，第 8 回社会システム部会研究会，計測自動制御学会（Mar. 2015）．

北澤正樹，高橋雅和，寺野隆雄：電子商取引を考慮した都市動態エージェントシミュレーション，第 63 回情報システム研究会，電気学会 電子・情報・システム部門（Jul. 2015）．

北澤正樹，吉川厚，寺野隆雄：サプライチェーン上の製造工場における部門連結を考慮した運用最適化，第 4 回経営課題に AI を! ビジネス・インフォマティクス研究会，人工知能学会（Mar. 2016）．

坂田顕庸，北澤正樹，高橋聡，國上真章，吉川厚，寺野隆雄：チーム問題解決における協調行動は MMORPG で発現するか？，エンターテイメントコンピューティング，情報処理学会，Vol. 2016-EC-41，No. 21，pp. 1-2（Jul. 2016）．

本松覚，北澤正樹，高橋聡，吉川厚，寺野隆雄：エージェントベースモデリングによる動的目標探索問題への接近，第 22 回創発システム・シンポジウム「創発夏の学校 2016」，計測自動制御学会，pp. 54（Aug. 2016）．

黄冬陽，相田晋，北澤正樹，高橋聡，高橋 B.徹，吉川厚，寺野隆雄：ウェアラブルセンサーを用いた機械学習による避難評価手法，第 12 回社会システム部会研究会，計測自動制御学会，pp. 52-61（Mar. 2017）（代理発表，最優秀発表賞）．

寺野隆雄，北澤正樹：ビーコンデバイスによる工場作業者の作業時間計測とシミュレーション"，第 173 回春季講演大会，鉄鋼協会，Vol. 30，pp. 26-27（Mar. 2017）（代理発表）．

特許

北澤正樹，江上慎，吉村知浩：製造プロセスの予測システムおよび予測制御システム，特願 2015-215416（2015/11/2），特開 2017-090947（2017/5/25），出願中.

付録

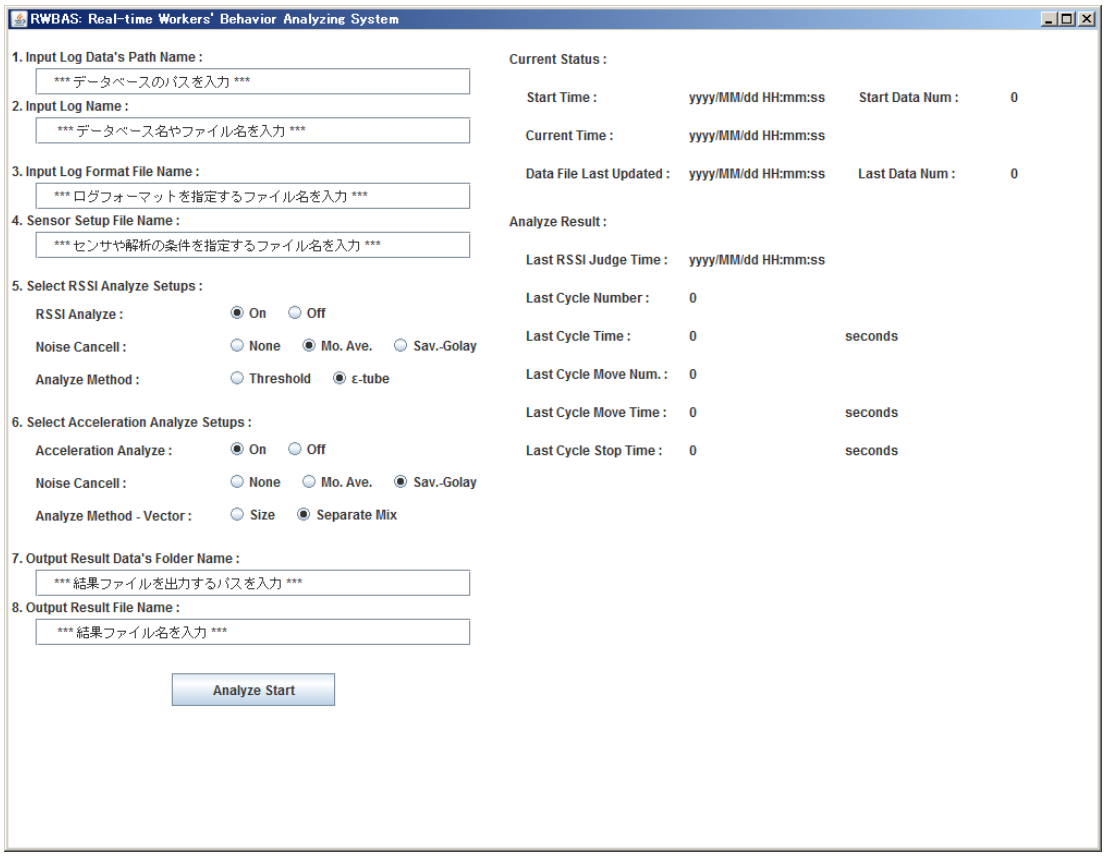
付録として本研究で構築したリアルタイム作業者動作計測システムのアプリ画面やソースコードの抜粋を掲載する．

付録 1. リアルタイム作業者動作計測システムのアプリ画面

作業者動作計測システムのアプリ画面は下図のようになっている．アプリの左半分は分析開始前に各種設定を入力する部分で，アプリの右半分はデータ状況や分析結果を表示する部分である．

アプリの左半分には 8 個の設定項目と 1 個のボタンがある．それぞれの内容は以下の通りである．

- 設定項目の 1 番目は作業者動作データを格納するデータベースやローカルファイルのパスを入力する項目である．
- 設定項目の 2 番目はデータベース名もしくはファイル名を入力する項目である．



図付録.1　リアルタイム作業者動作分析システムのアプリ画面

- 設定項目の 3 番目は作業者動作データのログフォーマットを指定するためのファイル名を入力する項目で，指定したファイル内では時刻や Beacon の RSSI や各加速度が何列目に格納されているかなどを指定する．
- 設定項目の 4 番目はウェアラブルセンサーの設定や分析パラメタを指定するためのファイル名を入力する項目で，指定したファイル内では各センサーの ID や各前処理や分析で使うデータ数や閾値を指定する．
- 設定項目の 5 番目は Beacon の RSSI データの前処理や分析手法を指定する項目で，ラジオボタンを押すことで変更が可能となっている．
- 設定項目の 6 番目は加速度データの前処理やベクトル成分分解の有無を指定する項目で，RSSI と同様にラジオボタンを押すことで変更が可能となっている．
- 設定項目の 7 番目は分析結果を出力するパスを入力する項目である．
- 設定項目の 8 番目は分析結果を出力するファイル名を入力する項目である．
- ボタンは設定項目を入力した後に分析を開始するために押すボタンで，押すとオン状態になり分析が開始される．分析を終了したい時はオン状態になっているボタンを再度押してオフ状態に切り替える．

アプリの右半分にはデータ状況として 3 項目と分析結果として 6 項目が表示される．それぞれの内容は以下のとおりである．

- データ状況の 1 番目は分析の開始時刻と開始時のデータベースのデータ数である．
- データ状況の 2 番目は現在の時刻である．
- データ状況の 3 番目はデータベースの最終更新時刻と最後に読み込んだ時のデータ数である．
- 分析結果の 1 番目は最後に Beacon の RSSI からピークとして抽出された時刻である製品 1 個の製造開始時刻である．
- 分析結果の 2 番目は分析を開始してから現在までに抽出したピークの数＝製品のサイクルタイムの数である．
- 分析結果の 3 番目は最後に計測した製品のサイクルタイムである．
- 分析結果の 4 番目は最後に計測した製品製造時の移動回数である．
- 分析結果の 5 番目は最後に計測した製品製造時の合計移動時間である．
- 分析結果の 6 番目は最後に計測した製品製造時の合計停止時間である主作業時間である．

付録 2. リアルタイム作業者動作計測システムのソースコード

構築したリアルタイム作業者動作計測システムのソースコードを記載する．なお，ログフォーマットとセンサー設定や分析パラメタを指定については，ファイルを読み込む別クラスが存在するが該当のクラス内で直接指定する形式で記載した．また，Servitzky-Golay filterはMarcin Rzeźnicki氏が作成したライブラリを活用しているので，使用時はインポートが必要である．

```
/*
 * RWBASmain : Real-time Workers' Behavior Analyzing System(RWBAS)のメインクラス
 * Masaki Kitazawa.
 *
 * RWBASのGUI設定や各種クラス（スレッド）の起動設定をするクラス。
 * データ読み込みなど、一部がローカルデータ使用版。
 */

import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import javax.swing.*;
import javax.swing.event.*;


public class RWBASmain extends JFrame {
   private static final long serialVersionUID = 1L;

   // Frame declaration
   static RWBASmain frame;
   static JLayeredPane layeredPane;
   static MenuFrame menuFrame;
   static ResultFrame resultFrame;

   // Frame Status
   int MenuX = 450;
   int ResultX = 550;
   int AllX = MenuX + ResultX;
   int AllY = 750;
   Color BGColor = Color.WHITE;

   // Input Flag
   static int ReadingFlag = 0;
   static int RSSIAnalyzingFlag = 0;
   static int RSSINoiseFlag = 0;
   static int RSSIMethodFlag = 0;
   static int AccelAnalyzingFlag = 0;
   static int AccelNoiseFlag = 0;
   static int AccelMethodVectorFlag = 0;

   // For MultiThread.
   // readLog(1) + RSSI(1) + Accel(1) + OtherProcess(1+ : Simulation et.al.)
   static int ThreadNum = 4;
      static ExecutorService service;

      // For Display Result
   static JLabel Start_SystemTime_L = new JLabel("yyyy/MM/dd HH:mm:ss");
   static JLabel Start_FinalLineNum_L = new JLabel("0");
   static JLabel LastCheck_FileUpdateTime_L = new JLabel("yyyy/MM/dd HH:mm:ss");
   static JLabel LastCheck_FinalLineNum_L = new JLabel("0");
   static JLabel Current_SystemTime_L = new JLabel("yyyy/MM/dd HH:mm:ss");
   static JLabel LastRSSIResult_Time_L = new JLabel("yyyy/MM/dd HH:mm:ss");
   static JLabel LastRSSIResult_CycleNum_L = new JLabel("0");
   static JLabel LastRSSIResult_CycleTime_L = new JLabel("0");
   static JLabel LastAccelResult_MoveNum_L = new JLabel("0");
   static JLabel LastAccelResult_MoveTime_L = new JLabel("0");
   static JLabel LastAccelResult_StopTime_L = new JLabel("0");

   // For Other Setups
   SimpleDateFormat DateFormat_Display = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
   SimpleDateFormat DateFormat_Data = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS");
   int InputErrorFlag = 0;
   int OutputErrorFlag = 0;

   // For Debug.
   static JLabel LogLabel;
   static JLabel LogLabel_Result;
   static JLabel TimeLabel;

   // Main Method
   public static void main(String args[]){
      frame = new RWBASmain("RWBAS: Real-time Workers' Behavior Analyzing System");
      frame.pack();

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
```

```
  // Anything Method
  RWBASmain(String title){
    setTitle(title);
      setLayout(new BorderLayout());
      layeredPane = new JLayeredPane();
      layeredPane.setPreferredSize(new Dimension(AllX,
AllY));
      add(layeredPane, BorderLayout.CENTER);

      // Setup for MenuFrame
      menuFrame = new MenuFrame();
      menuFrame.SetMenu();
      menuFrame.setLocation(0, 0);
      menuFrame.setSize(new Dimension(MenuX, AllY));
      layeredPane.add(menuFrame, new Integer(0));

      // Setup for ResultFrame - Read log
      resultFrame = new ResultFrame();
      resultFrame.SetMenu();
      resultFrame.setLocation(MenuX, 0);
      resultFrame.setSize(new Dimension(ResultX, AllY));
      layeredPane.add(resultFrame, new Integer(10));
  }


// Process of setting of menu frame.
  public class MenuFrame extends JPanel implements
ChangeListener, ActionListener {
    private static final long serialVersionUID = 1L;

    JTextField InputFolderText;
    JTextField InputDataFileText;
    JTextField FileFormatText;
    JTextField SensorSetupText;
    JTextField OutputFolderText;
    JTextField OutputFileText;
    JRadioButton[] RSSIOnOff = new JRadioButton[2];
    JRadioButton[] RSSINoise = new JRadioButton[3];
    JRadioButton[] RSSIAnalyze = new JRadioButton[2];
    JRadioButton[] AccelOnOff = new JRadioButton[2];
    JRadioButton[] AccelNoise = new JRadioButton[3];
    JRadioButton[] AccelAnalyze1 = new JRadioButton[2];
    JToggleButton AnalyzeStartButton = new
JToggleButton();

  // Setting of menu frame.
    public void SetMenu() {
      this.setBackground(BGColor);
        FlowLayout layout = (FlowLayout)this.getLayout();
        layout.setVgap(0);

        // Setup for Label & textField of "Folder Name"
        JLabel label = new JLabel(" 1. Input Log Data's
Path Name : ");
        label.setPreferredSize(new Dimension(MenuX,
35));
        label.setHorizontalAlignment(JLabel.LEFT);
        label.setVerticalAlignment(JLabel.BOTTOM);
        this.add(label);

        InputFolderText = new JTextField(36);
        this.add(InputFolderText);


        // Setup for Label & textField of "Log File"
        label = new JLabel(" 2. Input Log Name : ");
        label.setPreferredSize(new Dimension(MenuX,
20));
        label.setHorizontalAlignment(JLabel.LEFT);
        label.setVerticalAlignment(JLabel.BOTTOM);
        this.add(label);

        InputDataFileText = new JTextField(36);
        this.add(InputDataFileText);


        // Setup for Label & textField of "Format File"
        label = new JLabel(" 3. Input Log Format File
Name : ");
        label.setPreferredSize(new Dimension(MenuX,
35));
        label.setHorizontalAlignment(JLabel.LEFT);
        label.setVerticalAlignment(JLabel.BOTTOM);
        this.add(label);

        FileFormatText = new JTextField(36);
        this.add(FileFormatText);


        // Setup for Label & textField of "Format File"
        label = new JLabel(" 4. Sensor Setup File Name : ");
        label.setPreferredSize(new Dimension(MenuX,
20));
        label.setHorizontalAlignment(JLabel.LEFT);
        label.setVerticalAlignment(JLabel.BOTTOM);
        this.add(label);

        FileFormatText = new JTextField(36);
        this.add(FileFormatText);


        // Setup for Label & textField of "RSSI analyze
setups"
        label = new JLabel(" 5. Select RSSI Analyze
Setups : ");
        label.setPreferredSize(new Dimension(MenuX,
35));
        label.setHorizontalAlignment(JLabel.LEFT);
        label.setVerticalAlignment(JLabel.BOTTOM);
        this.add(label);

        JPanel tempPanel = new JPanel();
        tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
        tempPanel.setBackground(BGColor);
        tempPanel.setPreferredSize(new Dimension(MenuX,
30));
        label = new JLabel(" ");
        label.setPreferredSize(new Dimension(25, 30));
```

```
            tempPanel.add(label);
            label = new JLabel("RSSI Analyze : ");
            label.setPreferredSize(new Dimension(175, 30));
            tempPanel.add(label);
            RSSIOnOff[0] = new JRadioButton("On       ",
true);
            RSSIOnOff[1] = new JRadioButton("Off      ",
false);
            RSSIOnOff[0].setBackground(Color.WHITE);
            RSSIOnOff[1].setBackground(Color.WHITE);
            RSSIOnOff[0].addActionListener(this);
            RSSIOnOff[1].addActionListener(this);
            ButtonGroup RSSIOnOffGroup = new
ButtonGroup();
            RSSIOnOffGroup.add(RSSIOnOff[0]);
            RSSIOnOffGroup.add(RSSIOnOff[1]);
            tempPanel.add(RSSIOnOff[0]);
            tempPanel.add(RSSIOnOff[1]);
            this.add(tempPanel);

            tempPanel = new JPanel();
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new Dimension(MenuX,
30));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 30));
            tempPanel.add(label);
            label = new JLabel("Noise Cancell : ");
            label.setPreferredSize(new Dimension(175, 30));
            tempPanel.add(label);
            RSSINoise[0] = new JRadioButton("None     ",
false);
            RSSINoise[1] = new JRadioButton("Mo. Ave.     ",
true);
            RSSINoise[2] = new JRadioButton("Sav.-Golay
", false);
            RSSINoise[0].setBackground(Color.WHITE);
            RSSINoise[1].setBackground(Color.WHITE);
            RSSINoise[2].setBackground(Color.WHITE);
            RSSINoise[0].addActionListener(this);
            RSSINoise[1].addActionListener(this);
            RSSINoise[2].addActionListener(this);
            ButtonGroup RSSINoiseGroup = new
ButtonGroup();
            RSSINoiseGroup.add(RSSINoise[0]);
            RSSINoiseGroup.add(RSSINoise[1]);
            RSSINoiseGroup.add(RSSINoise[2]);
            tempPanel.add(RSSINoise[0]);
            tempPanel.add(RSSINoise[1]);
            tempPanel.add(RSSINoise[2]);
            this.add(tempPanel);

            tempPanel = new JPanel();
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new Dimension(MenuX,
30));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 30));
            tempPanel.add(label);
            label = new JLabel("Analyze Method : ");
            label.setPreferredSize(new Dimension(175, 30));
            tempPanel.add(label);
            RSSIAnalyze[0] = new JRadioButton("Threshold
", false);
            RSSIAnalyze[1] = new JRadioButton("ε-tube      ",
true);
            RSSIAnalyze[0].setBackground(Color.WHITE);
            RSSIAnalyze[1].setBackground(Color.WHITE);
            RSSIAnalyze[0].addActionListener(this);
            RSSIAnalyze[1].addActionListener(this);
            ButtonGroup RSSIAnalyzeGroup = new
ButtonGroup();
            RSSIAnalyzeGroup.add(RSSIAnalyze[0]);
            RSSIAnalyzeGroup.add(RSSIAnalyze[1]);
            tempPanel.add(RSSIAnalyze[0]);
            tempPanel.add(RSSIAnalyze[1]);
            this.add(tempPanel);


            // Setup for Label & textField of "RSSI analyze
setups"
            label = new JLabel(" 6. Select Acceleration Analyze
Setups : ");
            label.setPreferredSize(new Dimension(MenuX,
35));
            label.setHorizontalAlignment(JLabel.LEFT);
            label.setVerticalAlignment(JLabel.BOTTOM);
            this.add(label);

            tempPanel = new JPanel();
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new Dimension(MenuX,
30));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 30));
            tempPanel.add(label);
            label = new JLabel("Acceleration Analyze : ");
            label.setPreferredSize(new Dimension(175, 30));
            tempPanel.add(label);
            AccelOnOff[0] = new JRadioButton("On       ",
true);
            AccelOnOff[1] = new JRadioButton("Off      ",
false);
            AccelOnOff[0].setBackground(Color.WHITE);
            AccelOnOff[1].setBackground(Color.WHITE);
            AccelOnOff[0].addActionListener(this);
            AccelOnOff[1].addActionListener(this);
            ButtonGroup AccelOnOffGroup = new
ButtonGroup();
            AccelOnOffGroup.add(AccelOnOff[0]);
            AccelOnOffGroup.add(AccelOnOff[1]);
            tempPanel.add(AccelOnOff[0]);
            tempPanel.add(AccelOnOff[1]);
            this.add(tempPanel);
```

```
            tempPanel = new JPanel();
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new Dimension(MenuX,
30));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 30));
            tempPanel.add(label);
            label = new JLabel("Noise Cancell : ");
            label.setPreferredSize(new Dimension(175, 30));
            tempPanel.add(label);
            AccelNoise[0] = new JRadioButton("None       ",
false);
            AccelNoise[1] = new JRadioButton("Mo. Ave.
", false);
            AccelNoise[2] = new JRadioButton("Sav.-Golay
", true);
            AccelNoise[0].setBackground(Color.WHITE);
            AccelNoise[1].setBackground(Color.WHITE);
            AccelNoise[2].setBackground(Color.WHITE);
            AccelNoise[0].addActionListener(this);
            AccelNoise[1].addActionListener(this);
            AccelNoise[2].addActionListener(this);
            ButtonGroup AccelNoiseGroup = new
ButtonGroup();
            AccelNoiseGroup.add(AccelNoise[0]);
            AccelNoiseGroup.add(AccelNoise[1]);
            AccelNoiseGroup.add(AccelNoise[2]);
            tempPanel.add(AccelNoise[0]);
            tempPanel.add(AccelNoise[1]);
            tempPanel.add(AccelNoise[2]);
            this.add(tempPanel);

            tempPanel = new JPanel();
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new Dimension(MenuX,
30));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 30));
            tempPanel.add(label);
            label = new JLabel("Analyze Method - Vector : ");
            label.setPreferredSize(new Dimension(175, 30));
            tempPanel.add(label);
            AccelAnalyze1[0] = new JRadioButton("Size       ",
false);
            AccelAnalyze1[1] = new JRadioButton("Separate
Mix      ", true);
            AccelAnalyze1[0].setBackground(Color.WHITE);
            AccelAnalyze1[1].setBackground(Color.WHITE);
            AccelAnalyze1[0].addActionListener(this);
            AccelAnalyze1[1].addActionListener(this);
            ButtonGroup AccelAnalyzeGroup1 = new
ButtonGroup();
            AccelAnalyzeGroup1.add(AccelAnalyze1[0]);
            AccelAnalyzeGroup1.add(AccelAnalyze1[1]);
            tempPanel.add(AccelAnalyze1[0]);
            tempPanel.add(AccelAnalyze1[1]);
            this.add(tempPanel);


            // Setup for Label & textField of "Folder Name"
            label = new JLabel(" 7. Output Result Data's Folder
Name : ");
            label.setPreferredSize(new Dimension(MenuX,
35));
            label.setHorizontalAlignment(JLabel.LEFT);
            label.setVerticalAlignment(JLabel.BOTTOM);
            this.add(label);

            OutputFolderText = new JTextField(36);
            this.add(OutputFolderText);


            // Setup for Label & textField of "Log File"
            label = new JLabel(" 8. Output Result File Name :
");
            label.setPreferredSize(new Dimension(MenuX,
20));
            label.setHorizontalAlignment(JLabel.LEFT);
            label.setVerticalAlignment(JLabel.BOTTOM);
            this.add(label);

            OutputFileText = new JTextField(36);
            this.add(OutputFileText);


            // Setup for Label and Button of "Moving Pattern"
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(MenuX,
25));
            this.add(label);
            AnalyzeStartButton.setText(" Analyze Start ");
            AnalyzeStartButton.setPreferredSize(new
Dimension(150, 30));
            AnalyzeStartButton.addChangeListener(this);
            this.add(AnalyzeStartButton);

        }


    @Override
        public void actionPerformed(ActionEvent e) {
            // 押されるたびに全部読み込み
            for (int i = 0; i < RSSIOnOff.length; i++) {
                if (RSSIOnOff[i].isSelected()) {
                    RSSIAnalyzingFlag = i;
                    break;
                }
            }
            for (int i = 0; i < RSSINoise.length; i++) {
                if (RSSINoise[i].isSelected()) {
                    RSSINoiseFlag = i;
                    break;
                }
            }
            for (int i = 0; i < RSSIAnalyze.length; i++) {
```

```
            if (RSSIAnalyze[i].isSelected()) {
                RSSIMethodFlag = i;
                break;
            }
        }
        for (int i = 0; i < AccelOnOff.length; i++) {
            if (AccelOnOff[i].isSelected()) {
                AccelAnalyzingFlag = i;
                break;
            }
        }
        for (int i = 0; i < AccelNoise.length; i++) {
            if (AccelNoise[i].isSelected()) {
                AccelNoiseFlag = i;
                break;
            }
        }
        for (int i = 0; i < AccelAnalyze1.length; i++) {
            if (AccelAnalyze1[i].isSelected()) {
                AccelMethodVectorFlag = i;
                break;
            }
        }
    }


    // Process of input toggle button.
    @Override
    public void stateChanged(ChangeEvent e) {
        // Setting of toggle button.
        if (AnalyzeStartButton.isSelected()) {
            if (ReadingFlag == 0) {
                // Change Toggle Button.
                AnalyzeStartButton.setText("Realtime
Analyzing...");
                ReadingFlag = 1;

                // ボタンから分析条件取り込み
                for (int i = 0; i < RSSIOnOff.length; i++) {
                    if (RSSIOnOff[i].isSelected()) {
                        RSSIAnalyzingFlag = i;
                        break;
                    }
                }
                for (int i = 0; i < RSSINoise.length; i++) {
                    if (RSSINoise[i].isSelected()) {
                        RSSINoiseFlag = i;
                        break;
                    }
                }
                for (int i = 0; i < RSSIAnalyze.length; i++) {
                    if (RSSIAnalyze[i].isSelected()) {
                        RSSIMethodFlag = i;
                        break;
                    }
                }
                for (int i = 0; i < AccelOnOff.length; i++) {
                    if (AccelOnOff[i].isSelected()) {
                        AccelAnalyzingFlag = i;
                        break;
                    }
                }
                for (int i = 0; i < AccelNoise.length; i++) {
                    if (AccelNoise[i].isSelected()) {
                        AccelNoiseFlag = i;
                        break;
                    }
                }
                for (int i = 0; i < AccelAnalyze1.length; i++) {
                    if (AccelAnalyze1[i].isSelected()) {
                        AccelMethodVectorFlag = i;
                        break;
                    }
                }

                // Setup Radio Button are disabled
                for (int i = 0; i < RSSIOnOff.length; i++) {
                    RSSIOnOff[i].setEnabled(false);
                }
                for (int i = 0; i < RSSINoise.length; i++) {
                    RSSINoise[i].setEnabled(false);
                }
                for (int i = 0; i < RSSIAnalyze.length; i++) {
                    RSSIAnalyze[i].setEnabled(false);
                }
                for (int i = 0; i < AccelOnOff.length; i++) {
                    AccelOnOff[i].setEnabled(false);
                }
                for (int i = 0; i < AccelNoise.length; i++) {
                    AccelNoise[i].setEnabled(false);
                }
                for (int i = 0; i < AccelAnalyze1.length; i++) {
                    AccelAnalyze1[i].setEnabled(false);
                }

                // Get TextField String
                String LogFile;
                if (InputDataFileText.getText().length() == 0) {
                    InputDataFileText.setText("Error!! Input file
name here.");
                    LogFile = " ";
                    InputErrorFlag = 1;
                } else {
                    if (InputFolderText.getText().length() == 0) {
                        // ローカルデータ使用版
                        LogFile = "./" + InputDataFileText.getText()
+ ".csv";
                        InputErrorFlag = 0;
                    } else {
                        // ローカルデータ使用版
                        LogFile = InputFolderText.getText() +
InputDataFileText.getText() + ".csv";
                        InputErrorFlag = 0;
                    }
                }

                String ResultFile;
                if (OutputFileText.getText().length() == 0) {
                    OutputFileText.setText("Error!! Output file
name here.");
```

```
                ResultFile = " ";
                OutputErrorFlag = 1;
              } else {
                if (OutputFolderText.getText().length() == 0) {
                  ResultFile = "./" +
InputDataFileText.getText() + "_" + OutputFileText.getText();
                  OutputErrorFlag = 0;
                } else {
                  ResultFile = OutputFolderText.getText()
                      + InputDataFileText.getText() +
"_" + OutputFileText.getText();
                  OutputErrorFlag = 0;
                }
              }


              if (InputErrorFlag == 0 && OutputErrorFlag ==
0) {
                // Create Multi-Thread
                service =
Executors.newFixedThreadPool(ThreadNum);

                // Do not change submit order to setup first
status correctly.
                service.submit(new
RWBASotherprocess(ResultFile));
                service.submit(new
RWBASaccelanalyze(ResultFile));
                service.submit(new
RWBASrssianalyze(ResultFile));
                service.submit(new RWBASreadlog(LogFile));
              }
            }

          } else {
            if (ReadingFlag == 1) {
              // マルチスレッド終了
              service.shutdown();
            }

            // Setup Radio Button are enabled
            for (int i = 0; i < RSSIOnOff.length; i++) {
              RSSIOnOff[i].setEnabled(true);
            }
            for (int i = 0; i < RSSINoise.length; i++) {
              RSSINoise[i].setEnabled(true);
            }
            for (int i = 0; i < RSSIAnalyze.length; i++) {
              RSSIAnalyze[i].setEnabled(true);
            }
            for (int i = 0; i < AccelOnOff.length; i++) {
              AccelOnOff[i].setEnabled(true);
            }
            for (int i = 0; i < AccelNoise.length; i++) {
              AccelNoise[i].setEnabled(true);
            }
            for (int i = 0; i < AccelAnalyze1.length; i++) {
              AccelAnalyze1[i].setEnabled(true);
            }
            AnalyzeStartButton.setText(" Analyze Start ");
            ReadingFlag = 0;
          }
        }
      }


// Process of setting of menu frame.
  public class ResultFrame extends JPanel {
    private static final long serialVersionUID = 1L;

  // Display of result frame.
    public void SetMenu() {
      this.setBackground(Color.WHITE);
        FlowLayout layout = (FlowLayout)this.getLayout();
        layout.setVgap(0);

        JLabel label = new JLabel(" ");
        label.setPreferredSize(new Dimension(ResultX, 9));
        this.add(label);

        JPanel tempPanel = new JPanel();
        tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
        tempPanel.setBackground(BGColor);
        tempPanel.setPreferredSize(new
Dimension(ResultX, 35));
        label = new JLabel(" ");
        label.setPreferredSize(new Dimension(10, 35));
        tempPanel.add(label);
        label = new JLabel("Current Status : ");
        tempPanel.add(label);
        this.add(tempPanel);

        tempPanel = new JPanel();
        tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
        tempPanel.setBackground(BGColor);
        tempPanel.setPreferredSize(new
Dimension(ResultX, 35));
        label = new JLabel(" ");
        label.setPreferredSize(new Dimension(25, 35));
        tempPanel.add(label);
        label = new JLabel("Start Time : ");
        label.setPreferredSize(new Dimension(150, 35));
        tempPanel.add(label);
        Start_SystemTime_L.setPreferredSize(new
Dimension(140, 35));
        tempPanel.add(Start_SystemTime_L);
        label = new JLabel(" ");
        label.setPreferredSize(new Dimension(15, 35));
        tempPanel.add(label);
        label = new JLabel("Start Data Num : ");
        label.setPreferredSize(new Dimension(140, 35));
        tempPanel.add(label);
        tempPanel.add(Start_FinalLineNum_L);
        this.add(tempPanel);

        tempPanel = new JPanel();
        tempPanel.setLayout(new
```

```
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new
Dimension(ResultX, 35));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 35));
            tempPanel.add(label);
            label = new JLabel("Current Time : ");
            label.setPreferredSize(new Dimension(150, 35));
            tempPanel.add(label);
            Current_SystemTime_L.setPreferredSize(new
Dimension(140, 35));
            tempPanel.add(Current_SystemTime_L);
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(15, 35));
            tempPanel.add(label);
            this.add(tempPanel);

            tempPanel = new JPanel();
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new
Dimension(ResultX, 35));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 35));
            tempPanel.add(label);
            label = new JLabel("Data File Last Updated : ");
            label.setPreferredSize(new Dimension(150, 35));
            tempPanel.add(label);
            LastCheck_FileUpdateTime_L.setPreferredSize(new
Dimension(140, 35));
            tempPanel.add(LastCheck_FileUpdateTime_L);
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(15, 35));
            tempPanel.add(label);
            label = new JLabel("Last Data Num : ");
            label.setPreferredSize(new Dimension(140, 35));
            tempPanel.add(label);
            tempPanel.add(LastCheck_FinalLineNum_L);
            this.add(tempPanel);

            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(ResultX, 9));
            this.add(label);

            tempPanel = new JPanel();
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new
Dimension(ResultX, 35));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(10, 35));
            tempPanel.add(label);
            label = new JLabel("Analyze Result : ");
            tempPanel.add(label);
            this.add(tempPanel);

            tempPanel = new JPanel();
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new
Dimension(ResultX, 35));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 35));
            tempPanel.add(label);
            label = new JLabel("Last RSSI Judge Time : ");
            label.setPreferredSize(new Dimension(150, 35));
            tempPanel.add(label);
            LastRSSIResult_Time_L.setPreferredSize(new
Dimension(140, 35));
            tempPanel.add(LastRSSIResult_Time_L);
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(15, 35));
            tempPanel.add(label);
            this.add(tempPanel);

            tempPanel = new JPanel();
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new
Dimension(ResultX, 35));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 35));
            tempPanel.add(label);
            label = new JLabel("Last Cycle Number : ");
            label.setPreferredSize(new Dimension(150, 35));
            tempPanel.add(label);
            LastRSSIResult_CycleNum_L.setPreferredSize(new
Dimension(140, 35));
            tempPanel.add(LastRSSIResult_CycleNum_L);
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(15, 35));
            tempPanel.add(label);
            this.add(tempPanel);

            tempPanel = new JPanel();
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new
Dimension(ResultX, 35));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 35));
            tempPanel.add(label);
            label = new JLabel("Last Cycle Time : ");
            label.setPreferredSize(new Dimension(150, 35));
            tempPanel.add(label);
            LastRSSIResult_CycleTime_L.setPreferredSize(new
Dimension(140, 35));
            tempPanel.add(LastRSSIResult_CycleTime_L);
            label = new JLabel(" seconds ");
            label.setPreferredSize(new Dimension(80, 35));
            tempPanel.add(label);
            this.add(tempPanel);

            tempPanel = new JPanel();
```

```
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new
Dimension(ResultX, 35));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 35));
            tempPanel.add(label);
            label = new JLabel("Last Cycle Move Num. : ");
            label.setPreferredSize(new Dimension(150, 35));
            tempPanel.add(label);
            LastAccelResult_MoveNum_L.setPreferredSize(new
Dimension(140, 35));
            tempPanel.add(LastAccelResult_MoveNum_L);
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(80, 35));
            tempPanel.add(label);
            this.add(tempPanel);

            tempPanel = new JPanel();
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new
Dimension(ResultX, 35));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 35));
            tempPanel.add(label);
            label = new JLabel("Last Cycle Move Time : ");
            label.setPreferredSize(new Dimension(150, 35));
            tempPanel.add(label);
            LastAccelResult_MoveTime_L.setPreferredSize(new
Dimension(140, 35));
            tempPanel.add(LastAccelResult_MoveTime_L);
            label = new JLabel(" seconds ");
            label.setPreferredSize(new Dimension(80, 35));
            tempPanel.add(label);
            this.add(tempPanel);

            tempPanel = new JPanel();
            tempPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 0, 0));
            tempPanel.setBackground(BGColor);
            tempPanel.setPreferredSize(new
Dimension(ResultX, 35));
            label = new JLabel(" ");
            label.setPreferredSize(new Dimension(25, 35));
            tempPanel.add(label);
            label = new JLabel("Last Cycle Stop Time : ");
            label.setPreferredSize(new Dimension(150, 35));
            tempPanel.add(label);
            LastAccelResult_StopTime_L.setPreferredSize(new
Dimension(140, 35));
            tempPanel.add(LastAccelResult_StopTime_L);
            label = new JLabel(" seconds ");
            label.setPreferredSize(new Dimension(80, 35));
            tempPanel.add(label);
            this.add(tempPanel);

            LogLabel_Result = new JLabel("    ");
            LogLabel_Result.setHorizontalAlignment(JLabel.LEFT);
            this.add(LogLabel_Result);

        }
    }
}
```

ーーーーーーーーーーーーーーーーーーーークラスここまで.

```
/*
 * RWBASreadlog : RWBASのデータ読み込みクラス
 * Masaki Kitazawa.
 *
 * RWBASmainスレッドから投げられた"LogFile"変数で指定されたデータ位置を確認し、更新データを出力するクラス。
 * ローカル実験用のCSVファイルを検索しにくいバージョン。
 */

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.io.*;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;
import java.util.List;
import java.util.concurrent.Callable;
import javax.swing.*;


public class RWBASreadlog extends JPanel implements Callable<Integer> {
    private static final long serialVersionUID = 1L;

    // Transfer Variables
    String Start_SystemTime;
    JLabel Start_SystemTime_L;
    int Start_FinalLineNum;
    JLabel Start_FinalLineNum_L;
    String Current_SystemTime;
    JLabel Current_SystemTime_L;
    String LastCheck_FileUpdateTime;
    JLabel LastCheck_FileUpdateTime_L;
    int LastCheck_FinalLineNum;
    JLabel LastCheck_FinalLineNum_L;
    String LogFile;
    String LastRSSIResult_Time;
    JLabel LastRSSIResult_Time_L;
    int LastRSSIResult_FinalLineNum;

    // For Read Log
    String LastCheck_FinalDataLine;
    String LastRSSIResult_FinalDataLine;
    String Current_UpdateTime;
    String Current_FinalDataLine;

    static String Header = "Start";
    static List<String> TransDataLog = new ArrayList<String>();

    // For Other Setups
    SimpleDateFormat DateFormat_Display = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
    SimpleDateFormat DateFormat_Data = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'");
    int WaitFlag = 0;
    int LockFlag = 0;
    static int DataTransferFlag = 0;
    boolean StartTimeSaveFlag = true;
    Date DaStartDate;
    static long LoStartDate = 0;


// スレッド構築
    public RWBASreadlog(String LogFile) {

        // public -> local へ取り込み
        this.Start_SystemTime_L = RWBASmain.Start_SystemTime_L;
        this.Start_FinalLineNum_L = RWBASmain.Start_FinalLineNum_L;
        this.Current_SystemTime_L = RWBASmain.Current_SystemTime_L;
        this.LastCheck_FileUpdateTime_L = RWBASmain.LastCheck_FileUpdateTime_L;
        this.LastCheck_FinalLineNum_L = RWBASmain.LastCheck_FinalLineNum_L;
        this.LastRSSIResult_Time_L = RWBASmain.LastRSSIResult_Time_L;
        this.LogFile = LogFile;

    // ローカルデータ使用版、ここから
        // 読み込みデータの最終更新時刻を確認
        LastCheck_FileUpdateTime = DateFormat_Display.format(new File(LogFile).lastModified());

        // ファイルを読み込み
        try {
            FileReader fr = new FileReader(LogFile);
            BufferedReader br = new BufferedReader(fr);

            // 初期データ数確認
            String Newline = "";
            String line = "";
            int Count = 0;
            while ((Newline = br.readLine()) != null) {
                line = Newline;
                if (Count == 0) {
                    // 1列目はヘッダー格納
                    RWBASreadlog.Header = line;
                }
                Count = Count + 1;
            }

            if (Count == 0) {
                // ヘッダー含めてデータ無し時
                RWBASmain.ReadingFlag = 0;

            } else {
                // 初期設定
                this.Start_FinalLineNum = Count;
                this.LastCheck_FinalLineNum = Count;
                this.LastRSSIResult_FinalLineNum = Count;
                this.LastCheck_FinalDataLine = line;
```

```
                this.LastRSSIResult_FinalDataLine = line;
            }

            br.close();

        } catch (IOException ex) {
            //例外発生時処理
        }
    // ローカルデータ使用版、ここまで

        // 初期時刻設定
        if (this.LastRSSIResult_FinalLineNum == 1) {
            this.LastRSSIResult_Time = "1900/01/01 00:00:00";
        } else {
            String array[] =
this.LastRSSIResult_FinalDataLine.split( "," );
            try {
                this.LastRSSIResult_Time =
DateFormat_Display.format(DateFormat_Data.parse(array[2])
).toString();
            } catch (ParseException e) {
                e.printStackTrace();
            }
        }
        this.Start_SystemTime =
DateFormat_Display.format(new Date());

        // For Display

this.Start_SystemTime_L.setText(this.Start_SystemTime);

this.Start_FinalLineNum_L.setText(String.valueOf(Start_Fina
lLineNum));

this.LastCheck_FileUpdateTime_L.setText(LastCheck_FileU
pdateTime);

this.LastCheck_FinalLineNum_L.setText(String.valueOf(Last
Check_FinalLineNum));

this.LastRSSIResult_Time_L.setText(LastRSSIResult_Time);
    }


// メイン処理
    @Override
    public Integer call() throws Exception {
        while (RWBASmain.ReadingFlag == 1) {
            if (WaitFlag == 0) {
                if (DataTransferFlag == 0) {

                    // 現在時刻確認
                    this.Current_SystemTime =
DateFormat_Display.format(new Date());

this.Current_SystemTime_L.setText(Current_SystemTime);
                    Current_UpdateTime =
DateFormat_Display.format(new
File(LogFile).lastModified());   // ローカルデータ用
                    if
(Current_UpdateTime.equals(LastCheck_FileUpdateTime)) {

                    } else {
                        int Count = 0;

                        Path src = Paths.get(LogFile);
                    // ローカルデータ使用版、ここから
                    // データベース読み込み　&　更新データ格
納
                        Path TempFile =
Paths.get(".//LocalDataCopyFile.csv");
                        Files.copy(src, TempFile,
StandardCopyOption.COPY_ATTRIBUTES,
StandardCopyOption.REPLACE_EXISTING);

                        try {
                            // assign log file
                            FileReader fr = new
FileReader(TempFile.toString());
                            BufferedReader br = new
BufferedReader(fr);

                            // read log file (line by line).
                            String line = "";
                            while ((line = br.readLine()) != null) {
                                Count = Count + 1;
                                if (LastRSSIResult_FinalLineNum < 0) {
                                    if (Count > Start_FinalLineNum) {
                                        if (Count >
LastCheck_FinalLineNum) {
                                            TransDataLog.add(line);
                                        }
                                    }
                                } else {
                                    if (Count >
LastRSSIResult_FinalLineNum) {
                                        if (Count >
LastCheck_FinalLineNum) {
                                            TransDataLog.add(line);
                                        }
                                    }
                                }
                            }
                            br.close();

                        } catch (IOException ex) {
                                ex.printStackTrace();
                        }
                    // ローカルデータ使用版、ここまで

                        if (Count < Start_FinalLineNum) {
                            RWBASmain.ReadingFlag = 0;
                            break;

                        } else if (TransDataLog.size() <
(RWBASrssianalyze.PastMASize * 3)) {
                            // 念のため、データ数が多少たまるまで
待機
                            LastCheck_FileUpdateTime =
```

```
Current_UpdateTime;
                    LastCheck_FinalLineNum = Count;

this.LastCheck_FileUpdateTime_L.setText(LastCheck_FileU
pdateTime);

this.LastCheck_FinalLineNum_L.setText(String.valueOf(Last
Check_FinalLineNum));

                    repaint();
                    WaitFlag = 1;

                } else {
                    LastCheck_FileUpdateTime =
Current_UpdateTime;
                    LastCheck_FinalLineNum = Count;

this.LastCheck_FileUpdateTime_L.setText(LastCheck_FileU
pdateTime);

this.LastCheck_FinalLineNum_L.setText(String.valueOf(Last
Check_FinalLineNum));

                    repaint();
                    WaitFlag = 1;
                    DataTransferFlag = 1;

                }
              }
            } else {
              // RWBASrssianalyzeスレッド起動時の処理
              Thread.sleep(1000);
              this.Current_SystemTime =
DateFormat_Display.format(new Date());

this.Current_SystemTime_L.setText(Current_SystemTime);

            }

          } else {
            // 一度データを読み込んだ時は1秒は待機
            Thread.sleep(1000);
            WaitFlag = 0;

          }
        }

        return 0;
    }
}

――――――――――――――――――クラスここまで.
```

```
/*
 * RWBASrssianalyze : RWBASのRSSI分析クラス
 * Masaki Kitazawa.
 *
 * RWBASreadlogスレッドから受け渡されるデータより
Beaconデータを取り出し、
 * データ前処理、ピーク判定を行うことで製造進捗を計
測する。
 */


import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.concurrent.Callable;
import javax.swing.*;
import mr.go.sgfilter.ContinuousPadder;
import mr.go.sgfilter.SGFilter;


public class RWBASrssianalyze extends JPanel implements
Callable<Integer> {
    private static final long serialVersionUID = 1L;

// 変数宣言
    // Transfer Variables
    String LastRSSIResult_Time;
    JLabel LastRSSIResult_Time_L;
    int LastRSSIResult_FinalLineNum;
    int LastRSSIResult_CycleNum;
    JLabel LastRSSIResult_CycleNum_L;
    double LastRSSIResult_CycleTime;
    JLabel LastRSSIResult_CycleTime_L;

    // For Other Setups
    SimpleDateFormat DateFormat_Display = new
SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
    SimpleDateFormat DateFormat_Data = new
SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS");
    String ErrorLogLabelString;
    static String CycleResultFile;
    static int RSSIFinishFlag = 0;
    static int PastMASize;

    boolean RSSIMAcalc;
    boolean RSSISGcalc;
    boolean RSSIThr;
    boolean EpsTube;

    static List<Double> CycleTime = new
ArrayList<Double>();
    static List<String> CycleStartTime = new
ArrayList<String>();
    static List<String> CycleFinishTime = new
ArrayList<String>();
    List<Long> CalcTime = new ArrayList<Long>();
    long LastCycleStartTime = 0;
```

```
    int LastRSSIResult_LineNum = 0;
    Date LastRSSIResult_Time_Da;
    long LastRSSIResult_Time_lo;

    // RWBASaccelanalyzeスレッドへの受け渡し用データ格納変数
    static List<String> RSSI_TransDataLog;
    static String RSSI_LastCycleStartTime;
    static String RSSI_LastCycleFinishTime;

    // 次Cycle引き継ぎ用データ宣言
    // 正式にはLog Format Fileで指定した必要なデータ列番号のみ抽出する方法
    // 論文用にデータ形式を全部記載
    List<String> Trans_ObjectID = new ArrayList<String>();
    List<String> Trans_Detected = new ArrayList<String>();
    List<String> Trans_ReceivedTime = new ArrayList<String>();
    List<String> Trans_MacAd = new ArrayList<String>();
    List<Double> Trans_RSSI = new ArrayList<Double>();
    List<Double> Trans_UUID = new ArrayList<Double>();
    List<Double> Trans_MajorID = new ArrayList<Double>();
    List<Double> Trans_MinorID = new ArrayList<Double>();
    List<Double> Trans_MeasuredPower = new ArrayList<Double>();
    List<Double> Trans_Temperature = new ArrayList<Double>();
    List<Double> Trans_Voltage = new ArrayList<Double>();
    List<Double> Trans_AccelX = new ArrayList<Double>();
    List<Double> Trans_AccelY = new ArrayList<Double>();
    List<Double> Trans_AccelZ = new ArrayList<Double>();
    List<Double> Trans_Result = new ArrayList<Double>();
    List<Double> Trans_EtubeMAminusE = new ArrayList<Double>();
    List<Double> Trans_EtubeMA = new ArrayList<Double>();
    List<Double> Trans_EtubeMAplusE = new ArrayList<Double>();
    List<Double> Trans_PastMin = new ArrayList<Double>();
    List<Integer> Trans_EtubeList = new ArrayList<Integer>();
    List<Integer> Trans_MeasFlag = new ArrayList<Integer>();
    List<String> Trans_LatTime = new ArrayList<String>();
    List<Double> Trans_RssiResult = new ArrayList<Double>();
    List<Integer> Trans_Cluster = new ArrayList<Integer>();

//// 本来はスレッド構築時にフォーマットファイルなどから読み込む変数
//// 簡略化してスレッド内で宣言版
    // 胸センサーのID
    double MinorRSSIFilter = 『ID指定』;

    // 結果ファイル名をサイクルタイム用に補正。作業者名追加なども可
    String AddFileName = 『"_CycleTime.csv"など』;

    // MAFパラメタ
    int RSSIMAsize = 『データ値指定』;

    // SGFパラメタ
    int RSSIPartSize = 『データ数指定』;
    int RSSIDegree = 『近似多項式の次数指定』;

    // RSSI閾値パラメタ
    double RssiThreshold = 『閾値指定』;

    // Epsilon-tubeパラメタ
    int EpsPastMASize = 『データ数指定』;
    double Epsilon = 『εの幅指定』;
    double EpsMAbase = 『データ数未満時の値指定』;


// スレッド構築時の処理
    public RWBASrssianalyze(String ResultFile) {

        // public -> localへ取り込み
        this.LastRSSIResult_Time_L = RWBASmain.LastRSSIResult_Time_L;
        this.LastRSSIResult_CycleNum_L = RWBASmain.LastRSSIResult_CycleNum_L;
        this.LastRSSIResult_CycleTime_L = RWBASmain.LastRSSIResult_CycleTime_L;

        // first setup
        LastRSSIResult_Time = "1900/01/01 00:00:00.001";

this.LastRSSIResult_Time_L.setText(LastRSSIResult_Time);
        try {
            LastRSSIResult_Time_Da = DateFormat_Data.parse(LastRSSIResult_Time);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        LastRSSIResult_Time_lo = LastRSSIResult_Time_Da.getTime();

        LastRSSIResult_FinalLineNum = 0;
        LastRSSIResult_CycleNum = 0;

this.LastRSSIResult_CycleNum_L.setText(String.valueOf(LastRSSIResult_CycleNum));
        LastRSSIResult_CycleTime = 0.0;

this.LastRSSIResult_CycleTime_L.setText(String.valueOf(LastRSSIResult_CycleTime));

        // データ前処理手法の指定
        if (RWBASmain.RSSINoiseFlag == 1) {
            RSSIMAcalc = true;
            RSSISGcalc = false;
        } else if (RWBASmain.RSSINoiseFlag == 2) {
            RSSIMAcalc = false;
            RSSISGcalc = true;
        } else {
            RSSIMAcalc = false;
            RSSISGcalc = false;
        }
```

```
        // データ分析手法の指定
        if (RWBASmain.RSSIMethodFlag == 0) {
            RSSIThr = true;
            EpsTube = false;
        } else if (RWBASmain.RSSIMethodFlag == 1) {
            RSSIThr = false;
            EpsTube = true;
        } else {
            RSSIThr = false;
            EpsTube = false;
        }

        // 引き継ぎデータ数指定用
        if (RSSIMAcalc) {
            if (EpsTube) {
                if (RSSIMAsize < EpsPastMASize) {
                    PastMASize = EpsPastMASize;
                } else {
                    PastMASize = RSSIMAsize;
                }
            } else {
                PastMASize = RSSIMAsize;
            }
        } else {
            if (EpsTube) {
                if (RSSIPartSize < EpsPastMASize) {
                    PastMASize = EpsPastMASize;
                } else {
                    PastMASize = RSSIPartSize;
                }
            } else {
                PastMASize = RSSIPartSize;
            }
        }

        // 結果ファイル名をサイクルタイム用に補正
        RWBASrssianalyze.CycleResultFile = ResultFile +
AddFileName;
    }


// 処理開始
    @Override
    public Integer call() throws Exception {
        while (RWBASmain.ReadingFlag == 1) {
            if (RWBASreadlog.DataTransferFlag == 1) {
            // 分析のたびに初期化が必要な変数の宣言
                int DataSize = 0;
                int Count = 0;
                double TempValue;
                int MeasureFlag = 0;
                List<Integer> DataCount = new
ArrayList<Integer>();
                List<Integer> RSSICount = new
ArrayList<Integer>();
                int Local_LastRSSIResult_LineNum = 0;

                String LatestTime = "2016-01-01T00:00:00.000Z";
                SimpleDateFormat TempLaTime = new
SimpleDateFormat("yyyy-MM-DD'T'HH:mm:ss.SSS'Z'");
                Date DateLaTime = null;
                long LongLaTime;
                String CurrentTime = "2016-01-01T00:00:00.001Z";
                SimpleDateFormat TempCuTime = new
SimpleDateFormat("yyyy-MM-DD'T'HH:mm:ss.SSS'Z'");
                Date DateCuTime = null;
                long LongCuTime;
                int ClusterNum = 0;
                SimpleDateFormat LastDisplay = new
SimpleDateFormat("yyyy-MM-DD'T'HH:mm:ss.SSS'Z'");

                // Beaconデータ格納用宣言
                // 正式にはLog Format Fileで指定した必要なデ
ータ列番号のみ抽出
                // 論文用にデータ形式を全部記載
                List<String> ObjectID = new ArrayList<String>();
                List<String> Detected = new ArrayList<String>();
                List<String> ReceivedTime = new
ArrayList<String>();
                List<String> MacAd = new ArrayList<String>();
                List<Double> RSSI = new ArrayList<Double>();
                List<Double> UUID = new ArrayList<Double>();
                List<Double> MajorID = new
ArrayList<Double>();
                List<Double> MinorID = new
ArrayList<Double>();
                List<Double> MeasuredPower = new
ArrayList<Double>();
                List<Double> Temperature = new
ArrayList<Double>();
                List<Double> Voltage = new ArrayList<Double>();
                List<Double> AccelX = new ArrayList<Double>();
                List<Double> AccelY = new ArrayList<Double>();
                List<Double> AccelZ = new ArrayList<Double>();
                List<Double> EtubeMA = new
ArrayList<Double>();
                List<Double> EtubeMAplusE = new
ArrayList<Double>();
                List<Double> EtubeMAminusE = new
ArrayList<Double>();
                List<Double> PastMin = new ArrayList<Double>();
                List<Integer> EtubeList = new
ArrayList<Integer>();
                List<Integer> MeasFlag = new
ArrayList<Integer>();
                List<String> LatTime = new ArrayList<String>();
                List<Double> RssiResult = new
ArrayList<Double>();
                List<Integer> Cluster = new ArrayList<Integer>();

            // RWBASaccelanalyzeスレッド用に読み込んだデ
ータを格納
            // RWBASaccelanalyzeスレッドがデータを格納す
るまでは待機
                while (RSSIFinishFlag == 1) {
                    Thread.sleep(100);
                }
                RSSI_TransDataLog = new
ArrayList<String>(RWBASreadlog.TransDataLog);
```

```
        // 1列ずつデータ読み込み
            String line;
            Count = 0;
            while (Count <
RWBASreadlog.TransDataLog.size()) {
                line =
RWBASreadlog.TransDataLog.get(Count);
                String array[] = line.split( "," );

                    if (MinorRSSIFilter ==
Double.parseDouble(array[7])) {
                    // 胸のBeaconセンサーのデータのみ読
み込む
                        ObjectID.add(array[0]);
                        Detected.add(array[1]);
                    ReceivedTime.add(array[2]);
                    MacAd.add(array[3]);
                    RSSI.add(Double.parseDouble(array[4]));

UUID.add(Double.parseDouble(array[5]));

MajorID.add(Double.parseDouble(array[6]));

MinorID.add(Double.parseDouble(array[7]));

MeasuredPower.add(Double.parseDouble(array[8]));

Temperature.add(Double.parseDouble(array[9]));

Voltage.add(Double.parseDouble(array[10]));

AccelX.add(Double.parseDouble(array[11]));

AccelY.add(Double.parseDouble(array[12]));

AccelZ.add(Double.parseDouble(array[13]));

                    // 前サイクルからの引き継ぎデータ格
納
                    if (Trans_RSSI.isEmpty()) {
                        // 開始時は引き継ぎデータなし

                    } else {
                        // 前回からの引き継ぎデータを追加
                        if (RSSI.size() < (Trans_RSSI.size() -
PastMASize + 1)) {

EtubeMA.add(Trans_EtubeMA.get(RSSI.size() - 1));

EtubeMAplusE.add(Trans_EtubeMAplusE.get(RSSI.size() -
1));

EtubeMAminusE.add(Trans_EtubeMAminusE.get(RSSI.size()
- 1));

PastMin.add(Trans_PastMin.get(RSSI.size() - 1));

EtubeList.add(Trans_EtubeList.get(RSSI.size() - 1));

MeasFlag.add(Trans_MeasFlag.get(RSSI.size() - 1));

LatTime.add(Trans_LatTime.get(RSSI.size() - 1));

RssiResult.add(Trans_RssiResult.get(RSSI.size() - 1));

Cluster.add(Trans_Cluster.get(RSSI.size() - 1));
                        }
                    }
                    DataCount.add(Count + 1);
                    RSSICount.add(RSSI.size());

                } else {
                    // 胸のBeaconセンサーのデータでは
ないので読み飛ばし
                    DataCount.add(Count + 1);
                    RSSICount.add(9999);

                }
                Count = Count + 1;
            }


            DataSize = RSSI.size();
            // 引き継ぎデータなし、かつ、計測サイクルタ
イムもなし、の初期状態の場合
                if (Trans_RSSI.isEmpty() &&
CycleStartTime.isEmpty()) {
                    if (ReceivedTime.size() == 0) {
                        // 初期状態かつBeaconデータ無し

                    } else {
                        CurrentTime = ReceivedTime.get(0);
                        try {
                            DateCuTime =
TempCuTime.parse(CurrentTime);
                        } catch (ParseException e) {
                            e.printStackTrace();
                        }
                        LongCuTime = DateCuTime.getTime();
                        CycleStartTime.add(CurrentTime);
                        LastCycleStartTime = LongCuTime;
                    }
                } else if (CycleTime.isEmpty()) {

                } else {
                MeasureFlag = MeasFlag.get(MeasFlag.size() -
1);
                LatestTime = LatTime.get(LatTime.size() - 1);
                ClusterNum = Cluster.get(Cluster.size() - 1);
                }

            // RSSIデータ前処理
            // 処理結果格納用配列宣言
            double result[] = new double[DataSize];
            Arrays.fill(result, -9999.0);
            for (int i = 0; i < Trans_Result.size(); i++) {
                result[i] = Trans_Result.get(i);
            }

            // 移動平均を取る場合
```

```
        if (RSSIMAcalc) {
            for (int i = 0; i < DataSize; i++) {
                if (result[i] == -9999) {
                    if (i < RSSIMAsize) {
                        // データ位置が移動平均を計算する指
定数より手前の場合
                        TempValue = 0.0;
                        for (int j = 0; j < (i + RSSIMAsize + 1);
j++) {
                            TempValue = TempValue +
RSSI.get(j);
                        }
                        result[i] = TempValue / (i + RSSIMAsize
+ 1);

                    } else if (i < DataSize - RSSIMAsize) {
                        // 移動平均を計算する
                        TempValue = 0.0;
                        for (int j = i - RSSIMAsize; j < (i +
RSSIMAsize + 1); j++) {
                            TempValue = TempValue +
RSSI.get(j);
                        }
                        result[i] = TempValue / (2 * RSSIMAsize
+ 1);

                    } else {
                        // 移動平均を取れない最後を埋める
                        result[i] = -9999.0;
                    }
                }
            }
        }

        // SGFilterを使う場合
        if (RSSISGcalc) {
            // Marcin Rzeźnicki氏の公開されている
SGFilter.javaの使い方を踏襲
            double Coeffs[] = new double[DataSize];
            double data[] = new double[DataSize];

            for (int i = 0; i < DataSize; i++) {
                data[i] = RSSI.get(i);
            }

            Coeffs =
SGFilter.computeSGCoefficients(RSSIPartSize, RSSIPartSize,
RSSIDegree);
            ContinuousPadder padder1 = new
ContinuousPadder();
            SGFilter sgFilter = new SGFilter(RSSIPartSize,
RSSIPartSize);
            sgFilter.appendPreprocessor(padder1);
            result = sgFilter.smooth(data, Coeffs);
        }

        // MAもSGも使わない場合の計算
        if (RSSIMAcalc || RSSISGcalc) {

        } else {
            for (int i = 0; i < DataSize; i++) {
                result[i] = RSSI.get(i);
            }
        }

        // RSSIピーク判定
        // RSSIThreshold
        if (RSSIThr) {
            for (int i = 0; i < DataSize; i++) {
                if (i + 1 > RssiResult.size() ||
RssiResult.isEmpty()) {
                    if (i < DataSize - RSSIMAsize) {
                        // 1データずつ設定した閾値と比較し
て判定
                        if (result[i] > RssiThreshold) {
                            RssiResult.add(1.0);
                        } else {
                            RssiResult.add(0.0);
                        }
                    } else {
                        // 移動平均を取れない最後は判定しな
い
                        RssiResult.add(0.0);
                    }
                }

            }
        } else {
            // しきい値を使わない場合もリストを埋めて
おく
            for (int i = 0; i < DataSize; i++) {
                if (i + 1 > RssiResult.size() ||
RssiResult.isEmpty()) {
                    RssiResult.add(0.0);
                }

            }
        }

        // Epsilon-Tube法
        if (EpsTube) {
            for (int i = 0; i < DataSize; i++) {
                // Epsilon-Tubeの範囲決定
                // 前回からの引き継ぎデータで埋まってる
部分は飛ばす
                if (i + 1 > EtubeMA.size() ||
EtubeMA.isEmpty()) {
                    double TempPastMin = 0.0;

                    if (i < EpsPastMASize) {
                        EtubeMA.add(EpsMAbase);
                        EtubeMAplusE.add(EpsMAbase +
Epsilon);
                        EtubeMAminusE.add(EpsMAbase -
Epsilon);

                    int PastCount = 0;
                    for (int j = 0; j < i; j++) {
                        if (result[j] < TempPastMin) {
                            TempPastMin = result[j];
```

```
                    }
                    PastCount = PastCount + 1;
                }
                if (EpsMAbase < TempPastMin ) {
                    TempPastMin = EpsMAbase;
                }

            } else {
                if (i < DataSize - RSSIMAsize) {
                    // Epsilon-tubeの中央の移動平均計算
                    TempValue = 0.0;
                    for (int j = i - EpsPastMASize; j < i;
j++) {
                        TempValue = TempValue + result[j];
                    }
                    TempValue = TempValue /
EpsPastMASize;

                    // 閾値設定
                    EtubeMA.add(TempValue);
                    EtubeMAplusE.add(TempValue +
Epsilon);
                    EtubeMAminusE.add(TempValue -
Epsilon);

                    int PastCount = 0;
                    for (int j = i - EpsPastMASize; j < i -
(EpsPastMASize / 2); j++) {
                        if (result[j] < TempPastMin) {
                            TempPastMin = result[j];
                        }
                        PastCount = PastCount + 1;
                    }
                } else {
                    EtubeMA.add(0.0);
                    EtubeMAplusE.add(999.0);
                    EtubeMAminusE.add(-999.0);
                }
            }

            // 判定
            if (i < DataSize - RSSIMAsize) {
                if (result[i] > EtubeMAplusE.get(i)) {
                    EtubeList.add(1);
                } else {
                    EtubeList.add(0);
                }
            } else {
                EtubeList.add(0);
            }

            PastMin.add(TempPastMin);
        }
    }
} else {
    // Epsilon-tube法を使わない場合もリストを埋
めておく
    for (int i = 0; i < DataSize; i++) {
        if (i + 1 > EtubeMA.size() ||
EtubeMA.isEmpty()) {
            EtubeMA.add(9999.0);
            EtubeMAplusE.add(99999.0);
            EtubeMAminusE.add(999.0);
            PastMin.add(99.0);
            EtubeList.add(0);
        }
    }
}

// 念のため閾値法もEpsTube法も選ばれなかっ
た場合
if (RSSIThr || EpsTube) {

} else {
    System.out.println("RSSI Judge is not
selecting!!");
}

// RSSIのクラスター作成および時間抽出
for (int i = 0; i < DataSize - RSSIMAsize + 1; i++)
{

    CurrentTime = ReceivedTime.get(i);
    try {
        DateCuTime =
TempCuTime.parse(CurrentTime);
    } catch (ParseException e) {
        e.printStackTrace();
    }
    LongCuTime = DateCuTime.getTime();

    if (i + 1 > Cluster.size() || Cluster.isEmpty()) {

        if (MeasureFlag == 2) {
            if (EtubeList.get(i) == 1 || RssiResult.get(i)
== 1) {
                // 継続のためLatest更新のみ
                MeasFlag.add(MeasureFlag);
                Cluster.add(ClusterNum);
                LatestTime = CurrentTime;
                LatTime.add(LatestTime);
            } else {
                MeasureFlag = 0;
                MeasFlag.add(MeasureFlag);
                LatTime.add(LatestTime);
                Cluster.add(ClusterNum);
            }

        } else if (MeasureFlag == 1) {
            if (EtubeList.get(i) == 1 || RssiResult.get(i)
== 1) {
                // 継続のためLatest更新のみ
                MeasFlag.add(MeasureFlag);
                Cluster.add(ClusterNum);
                LatestTime = CurrentTime;
                LatTime.add(LatestTime);
            } else {
                MeasureFlag = 0;
                MeasFlag.add(MeasureFlag);
                LatTime.add(LatestTime);
```

```
                    Cluster.add(ClusterNum);
                }

            } else {
                if (EtubeList.get(i) >= 1 || RssiResult.get(i)
>= 1) {
                    // 新たなピーク認定
                    try {
                        DateLaTime =
TempLaTime.parse(LatestTime);
                    } catch (ParseException e) {
                        e.printStackTrace();
                    }
                    LongLaTime = DateLaTime.getTime();

                    if (CycleFinishTime.size() <=
ClusterNum) {
                        CycleFinishTime.add(CurrentTime);
                        TempValue = (int)(LongCuTime -
LastCycleStartTime);
                        TempValue = (Double)(TempValue /
1000);
                        CycleTime.add(TempValue);
                        CycleStartTime.add(CurrentTime);
                    }
                    LastCycleStartTime = LongCuTime;
                    Local_LastRSSIResult_LineNum = i;
                    LastRSSIResult_LineNum =
DataCount.get(RSSICount.indexOf(i+1));
                    ClusterNum = ClusterNum + 1;
                    Cluster.add(ClusterNum);
                    MeasureFlag = 1;
                    MeasFlag.add(MeasureFlag);
                    LatestTime = CurrentTime;
                    LatTime.add(LatestTime);

                } else {
                    MeasFlag.add(MeasureFlag);
                    LatTime.add(LatestTime);
                    Cluster.add(ClusterNum);
                }
            }
        }
        for (int i = DataSize - RSSIMAsize + 1; i <
DataSize; i++) {
            MeasFlag.add(MeasureFlag);
            LatTime.add(LatestTime);
            Cluster.add(ClusterNum);
        }

    // 結果処理
        if (CycleFinishTime.isEmpty()) {
            // サイクルタイムなしの場合

        } else if
(LastRSSIResult_Time.equals(CycleFinishTime.get(CycleFini
shTime.size() - 1))) {
            // サイクルタイム更新なしの場合
        } else {
            // サイクルタイム更新の場合

            // GUI更新
            LastRSSIResult_Time =
CycleFinishTime.get(CycleFinishTime.size() - 1);
            try {
                Date date =
LastDisplay.parse(LastRSSIResult_Time);
                LastDisplay.applyPattern("yyyy/MM/dd
HH:mm:ss");

this.LastRSSIResult_Time_L.setText(LastDisplay.format(date
).toString());

                LastRSSIResult_Time_Da =
TempLaTime.parse(LastRSSIResult_Time);
            } catch (ParseException e) {
                e.printStackTrace();
            }
            LastRSSIResult_Time_lo =
LastRSSIResult_Time_Da.getTime();
            LastRSSIResult_FinalLineNum =
LastRSSIResult_FinalLineNum + LastRSSIResult_LineNum;
            LastRSSIResult_CycleNum =
Cluster.get(Cluster.size() - 1);

this.LastRSSIResult_CycleNum_L.setText(String.valueOf(La
stRSSIResult_CycleNum));
            LastRSSIResult_CycleTime =
CycleTime.get(CycleTime.size() - 1);

this.LastRSSIResult_CycleTime_L.setText(String.valueOf(La
stRSSIResult_CycleTime));
            repaint();

            // 次回分析への引き継ぎデータ準備；　前デ
ータ削除
            Trans_ObjectID.clear();
            Trans_Detected.clear();
            Trans_ReceivedTime.clear();
            Trans_MacAd.clear();
            Trans_RSSI.clear();
            Trans_UUID.clear();
            Trans_MajorID.clear();
            Trans_MinorID.clear();
            Trans_MeasuredPower.clear();
            Trans_Temperature.clear();
            Trans_Voltage.clear();
            Trans_AccelX.clear();
            Trans_AccelY.clear();
            Trans_AccelZ.clear();
            Trans_Result.clear();
            Trans_EtubeMAminusE.clear();
            Trans_EtubeMA.clear();
            Trans_EtubeMAplusE.clear();
            Trans_PastMin.clear();
            Trans_EtubeList.clear();
            Trans_MeasFlag.clear();
                Trans_LatTime.clear();
```

```
                Trans_RssiResult.clear();
                Trans_Cluster.clear();

                // 次回分析への引き継ぎデータ準備; 引き継ぎ開始位置設定
                if (Local_LastRSSIResult_LineNum - (PastMASize * 2) < 0) {
                  TempValue = 0;
                } else {
                  TempValue = Local_LastRSSIResult_LineNum - (PastMASize * 2);
                }

                // 次回分析への引き継ぎデータ準備; 格納
              for (int i = (int)TempValue; i < DataSize; i++) {
                Trans_ObjectID.add(ObjectID.get(i));
                Trans_Detected.add(Detected.get(i));

Trans_ReceivedTime.add(ReceivedTime.get(i));
                Trans_MacAd.add(MacAd.get(i));
                Trans_RSSI.add(RSSI.get(i));
                Trans_UUID.add(UUID.get(i));
                Trans_MajorID.add(MajorID.get(i));
                Trans_MinorID.add(MinorID.get(i));

Trans_MeasuredPower.add(MeasuredPower.get(i));
                Trans_Temperature.add(Temperature.get(i));
                Trans_Voltage.add(Voltage.get(i));
                Trans_AccelX.add(AccelX.get(i));
                Trans_AccelY.add(AccelY.get(i));
                Trans_AccelZ.add(AccelZ.get(i));
                Trans_Result.add(result[i]);

Trans_EtubeMAminusE.add(EtubeMAminusE.get(i));
                Trans_EtubeMA.add(EtubeMA.get(i));

Trans_EtubeMAplusE.add(EtubeMAplusE.get(i));
                Trans_PastMin.add(PastMin.get(i));
                Trans_EtubeList.add(EtubeList.get(i));
                Trans_MeasFlag.add(MeasFlag.get(i));
                Trans_LatTime.add(LatTime.get(i));
                Trans_RssiResult.add(RssiResult.get(i));
                Trans_Cluster.add(Cluster.get(i));
              }

              // RWBASreadlogからの移動データ消去: 消去終了位置設定
                if (Local_LastRSSIResult_LineNum - (PastMASize * 2) < 1) {
                  TempValue = 1;
                } else {
                  TempValue = Local_LastRSSIResult_LineNum - (PastMASize * 2);
                }

                // RWBASreadlogスレッドからの移動データ消去: 消去実施
              for (int i = 0; i < DataCount.get(RSSICount.indexOf((int)TempValue)); i++) {
                RWBASreadlog.TransDataLog.remove(0);
              }

            // RWBASaccelanalyzeスレッドの開始設定
              RSSI_LastCycleStartTime = CycleStartTime.get(CycleStartTime.size() - 2);
              RSSI_LastCycleFinishTime = CycleFinishTime.get(CycleFinishTime.size() - 1);
              RSSIFinishFlag = 1;

            }

            // 新サイクルタイムの有無に関わらず 1 度処理したら再度データ読み込み可能にする
            RWBASreadlog.DataTransferFlag = 0;

          } else {
            // RWBASreadlogからのデータがないので待機
            Thread.sleep(1000);

          }

        }

        // 終了処理
        return 0;
      }

}

――――――――――――――――――クラスここまで.
```

```
/*
 * RWBASaccelanalyze : RWBASの加速度分析クラス
 * Masaki Kitazawa.
 *
 * RWBASrssianalyzeスレッドから受け渡されるデータより加速度データを取り出し、
 * データ前処理、ベクトル分解、閾値判定を行うことで作業者の移動・停止のタイミングを推定し、
 * 作業履歴と主作業時間を計測する。
 */


import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.concurrent.Callable;
import javax.swing.*;
import mr.go.sgfilter.ContinuousPadder;
import mr.go.sgfilter.SGFilter;


public class RWBASaccelanalyze extends JPanel
implements Callable<Integer> {
    private static final long serialVersionUID = 1L;

//  変数宣言
    // Transfer Variables
    int LastAccelResult_MoveNum;
    JLabel LastAccelResult_MoveNum_L;
    double LastAccelResult_MoveTime;
    JLabel LastAccelResult_MoveTime_L;
    double LastAccelResult_StopTime;
    JLabel LastAccelResult_StopTime_L;

    // For Other Setups
    static String AccelResultFile;
    static List<Integer> AccelMoveNum = new
ArrayList<Integer>();
    static List<Double> AccelMoveTime = new
ArrayList<Double>();
    static List<Double> AccelStopTime = new
ArrayList<Double>();
    static List<Double> AccelCycleTime;
    static int AccelFinishFlag = 0;

    boolean AccelMAcalc;
    boolean AccelSGcalc;
    boolean AccelSeparate;
    String LastCycleStartTime;
    SimpleDateFormat TempLaCyStTime = new
SimpleDateFormat("yyyy-MM-DD'T'HH:mm:ss.SSS'Z'");
    Date DateLaCyStTime = null;
    long LongLaCyStTime;
    String LastCycleFinishTime;
    SimpleDateFormat TempLaCyFiTime = new
SimpleDateFormat("yyyy-MM-DD'T'HH:mm:ss.SSS'Z'");
    Date DateLaCyFiTime = null;
    long LongLaCyFiTime;

////本当はスレッド構築時にフォーマットファイルなどから読み込む変数
////論文用にスレッド内で宣言
    // 腰センサーのID
    double MinorHipAccelFilter = 『ID指定』;

    // 足首センサーのID
    double MinorAnkleAccelFilter = 『ID指定』;

    // 結果ファイル名をサイクルタイム用に補正。作業者名追加なども可
    String AddFileName = 『_AccelResult.csvなど』;

    // MAFのパラメタ
    int AccelMAsize = 『データ数指定』;

    // SGFのパラメタ
    int AccelPartSize = 『データ数指定』;
    int AccelDegree = 『近似多項式の次数指定』;

    // 静止時の腰の重力方向ベクトルの成分
    // センサXYZを固定できる場合は準備値、固定できない場合は事前測定値を設定
    double HipAccelXAve = 『X成分』;
    double HipAccelYAve = 『Y成分』;
    double HipAccelZAve = 『Z成分』;
    double HipAccelXYZ =
Math.sqrt(Math.pow(HipAccelXAve, 2) +
Math.pow(HipAccelYAve, 2) + Math.pow(HipAccelZAve,
2));

    // 静止時の足首の重力方向ベクトルの成分
    // センサXYZを固定できる場合は準備値、固定できない場合は事前測定値を設定
    double AnkleAccelXAve = 『X成分』;
    double AnkleAccelYAve = 『Y成分』;
    double AnkleAccelZAve = 『Z成分』;

    // 腰センサの躍度の閾値
    double HipJerkThreshold = 『閾値』;

    // 足首センサの躍度の閾値
    double AnkleJerkThreshold = 『閾値』;


// スレッド構築時の処理
    public RWBASaccelanalyze(String ResultFile) {

        // public -> localへ取り込み
        this.LastAccelResult_MoveNum_L =
RWBASmain.LastAccelResult_MoveNum_L;
        this.LastAccelResult_MoveTime_L =
RWBASmain.LastAccelResult_MoveTime_L;
        this.LastAccelResult_StopTime_L =
RWBASmain.LastAccelResult_StopTime_L;

        // first setup
```

```
        LastAccelResult_MoveNum = 0;

this.LastAccelResult_MoveNum_L.setText(String.valueOf(LastAccelResult_MoveNum));
        LastAccelResult_MoveTime = 0.0;

this.LastAccelResult_MoveTime_L.setText(String.valueOf(LastAccelResult_MoveTime));
        LastAccelResult_StopTime = 0.0;

this.LastAccelResult_StopTime_L.setText(String.valueOf(LastAccelResult_StopTime));

        // データ前処理手法の指定
        if (RWBASmain.AccelNoiseFlag == 1) {
            AccelMAcalc = true;
            AccelSGcalc = false;
        } else if (RWBASmain.AccelNoiseFlag == 2) {
            AccelMAcalc = false;
            AccelSGcalc = true;
        } else {
            AccelMAcalc = false;
            AccelSGcalc = false;
        }

        // ベクトル分解有無の指定
        if (RWBASmain.AccelMethodVectorFlag == 0) {
            AccelSeparate = false;
        } else {
            AccelSeparate = true;
        }

        // 結果ファイル名をサイクルタイム用に補正
        RWBASaccelanalyze.AccelResultFile = ResultFile + AddFileName;
    }


// メイン処理
    @Override
    public Integer call() throws Exception {
        while (RWBASmain.ReadingFlag == 1) {
            if (RWBASrssianalyze.RSSIFinishFlag == 1) {
            // 分析のたびに初期化が必要な変数の宣言
                int Count = 0;
                int Count2 = 0;
                int HipDataSize = 0;
                int AnkleDataSize = 0;
                double TempValue, TempValue2;
                double TempX, TempY, TempZ;
                List<Double> TempAccelMoveTime = new ArrayList<Double>();

                String CurrentTime = "2016-01-01T00:00:00.001Z";
                SimpleDateFormat TempCuTime = new SimpleDateFormat("yyyy-MM-DD'T'HH:mm:ss.SSS'Z'");
                Date DateCuTime = null;
                long LongCuTime;
                double DefTime = 1.0;
                double DefAccel = 1.0;

                // スレッド内での分割格納用の変数宣言
                // 正式にはLog Format Fileで指定した必要なデータ列番号のみ抽出する
                // 論文用にデータ形式を全部記載
                // 腰加速度
                List<String> HipObjectID = new ArrayList<String>();
                List<String> HipDetected = new ArrayList<String>();
                List<String> HipReceivedTime = new ArrayList<String>();
                List<String> HipMacAd = new ArrayList<String>();
                List<Double> HipRSSI = new ArrayList<Double>();
                List<Double> HipUUID = new ArrayList<Double>();
                List<Double> HipMajorID = new ArrayList<Double>();
                List<Double> HipMinorID = new ArrayList<Double>();
                List<Double> HipMeasuredPower = new ArrayList<Double>();
                List<Double> HipTemperature = new ArrayList<Double>();
                List<Double> HipVoltage = new ArrayList<Double>();
                List<Double> HipAccelX = new ArrayList<Double>();
                List<Double> HipAccelY = new ArrayList<Double>();
                List<Double> HipAccelZ = new ArrayList<Double>();
                List<Double> HipVirtAccelSize = new ArrayList<Double>();
                List<Double> HipHoriAccelSize = new ArrayList<Double>();
                List<Double> HipAccelSize = new ArrayList<Double>();
                List<Double> HipAccelSizeNC = new ArrayList<Double>();
                List<Double> HipJerk = new ArrayList<Double>();
                List<Double> HipJerkAbs = new ArrayList<Double>();
                List<Double> HipAccelJudge = new ArrayList<Double>();
                List<Long> HipLongTime = new ArrayList<Long>();

                // 足首加速度
                List<String> AnkleObjectID = new ArrayList<String>();
                List<String> AnkleDetected = new ArrayList<String>();
                List<String> AnkleReceivedTime = new ArrayList<String>();
                List<String> AnkleMacAd = new ArrayList<String>();
                List<Double> AnkleRSSI = new
```

```
ArrayList<Double>();
            List<Double> AnkleUUID = new
ArrayList<Double>();
            List<Double> AnkleMajorID = new
ArrayList<Double>();
            List<Double> AnkleMinorID = new
ArrayList<Double>();
            List<Double> AnkleMeasuredPower = new
ArrayList<Double>();
            List<Double> AnkleTemperature = new
ArrayList<Double>();
            List<Double> AnkleVoltage = new
ArrayList<Double>();
            List<Double> AnkleAccelX = new
ArrayList<Double>();
            List<Double> AnkleAccelY = new
ArrayList<Double>();
            List<Double> AnkleAccelZ = new
ArrayList<Double>();
            List<Double> AnkleAccelSize = new
ArrayList<Double>();
            List<Double> AnkleAccelSizeNC = new
ArrayList<Double>();
            List<Double> AnkleJerk = new
ArrayList<Double>();
            List<Double> AnkleJerkAbs = new
ArrayList<Double>();
            List<Double> AnkleAccelJudge = new
ArrayList<Double>();
            List<Long> AnkleLongTime = new
ArrayList<Long>();

            // 合算判定用
            List<String> AccelObjectID = new
ArrayList<String>();
            List<String> AccelDetected = new
ArrayList<String>();
            List<String> AccelReceivedTime = new
ArrayList<String>();
            List<String> AccelMacAd = new
ArrayList<String>();
            List<Double> AccelRSSI = new
ArrayList<Double>();
            List<Double> AccelUUID = new
ArrayList<Double>();
            List<Double> AccelMajorID = new
ArrayList<Double>();
            List<Double> AccelMinorID = new
ArrayList<Double>();
            List<Double> AccelMeasuredPower = new
ArrayList<Double>();
            List<Double> AccelTemperature = new
ArrayList<Double>();
            List<Double> AccelVoltage = new
ArrayList<Double>();
            List<Double> AccelAccelX = new
ArrayList<Double>();
            List<Double> AccelAccelY = new
ArrayList<Double>();
            List<Double> AccelAccelZ = new
ArrayList<Double>();
            List<Double> AccelVirtAccelSize = new
ArrayList<Double>();
            List<Double> AccelHoriAccelSize = new
ArrayList<Double>();
            List<Double> AccelAccelSize = new
ArrayList<Double>();
            List<Double> AccelAccelSizeNC = new
ArrayList<Double>();
            List<Double> AccelJerk = new
ArrayList<Double>();
            List<Double> AccelJerkAbs = new
ArrayList<Double>();
            List<Double> AccelAccelJudge = new
ArrayList<Double>();
            List<Long> AccelLongTime = new
ArrayList<Long>();
            List<Integer> AccelCluster = new
ArrayList<Integer>();


        // データ読み込み
            String line;
            Count = 0;
            while (Count <
RWBASrssianalyze.RSSI_TransDataLog.size()) {
                line =
RWBASrssianalyze.RSSI_TransDataLog.get(Count);
                String array[] = line.split( "," );

                    if (MinorHipAccelFilter ==
Double.parseDouble(array[7])) {
                        // 腰センサのデータ格納
                        HipObjectID.add(array[0]);
                        HipDetected.add(array[1]);
                        HipReceivedTime.add(array[2]);
                        HipMacAd.add(array[3]);

HipRSSI.add(Double.parseDouble(array[4]));

HipUUID.add(Double.parseDouble(array[5]));

HipMajorID.add(Double.parseDouble(array[6]));

HipMinorID.add(Double.parseDouble(array[7]));

HipMeasuredPower.add(Double.parseDouble(array[8]));

HipTemperature.add(Double.parseDouble(array[9]));

HipVoltage.add(Double.parseDouble(array[10]));

HipAccelX.add(Double.parseDouble(array[11]));

HipAccelY.add(Double.parseDouble(array[12]));

HipAccelZ.add(Double.parseDouble(array[13]));

                    } else if (MinorAnkleAccelFilter ==
Double.parseDouble(array[7])){
```

```
                    // 足首センサのデータ格納
                    AnkleObjectID.add(array[0]);
                    AnkleDetected.add(array[1]);
                    AnkleReceivedTime.add(array[2]);
                    AnkleMacAd.add(array[3]);

AnkleRSSI.add(Double.parseDouble(array[4]));

AnkleUUID.add(Double.parseDouble(array[5]));

AnkleMajorID.add(Double.parseDouble(array[6]));

AnkleMinorID.add(Double.parseDouble(array[7]));

AnkleMeasuredPower.add(Double.parseDouble(array[8]));

AnkleTemperature.add(Double.parseDouble(array[9]));

AnkleVoltage.add(Double.parseDouble(array[10]));

AnkleAccelX.add(Double.parseDouble(array[11]));

AnkleAccelY.add(Double.parseDouble(array[12]));

AnkleAccelZ.add(Double.parseDouble(array[13]));
                }

                Count = Count + 1;
            }


        // RWBASrssianalyzeスレッドからの値引き継ぎ
            AccelCycleTime = new
ArrayList<Double>(RWBASrssianalyze.CycleTime);

            this.LastCycleStartTime =
RWBASrssianalyze.RSSI_LastCycleStartTime;
            this.LastCycleFinishTime =
RWBASrssianalyze.RSSI_LastCycleFinishTime;
            try {
                DateLaCyStTime =
TempLaCyStTime.parse(LastCycleStartTime);
                DateLaCyFiTime =
TempLaCyStTime.parse(LastCycleFinishTime);
            } catch (ParseException e) {
                e.printStackTrace();
            }
            LongLaCyStTime = DateLaCyStTime.getTime();
            LongLaCyFiTime = DateLaCyFiTime.getTime();

        // RWBASrssianalyzeスレッドのフラグ開放
            RWBASrssianalyze.RSSIFinishFlag = 0;
            RWBASrssianalyze.RSSI_TransDataLog.clear();


        // 時刻計算やベクトル分解の実施
            HipDataSize = HipRSSI.size();
            AnkleDataSize = AnkleRSSI.size();

                // 腰センサ
            for (int i = 0; i < HipAccelX.size(); i++) {
                // データの時刻をLongに変換し格納
                CurrentTime = HipReceivedTime.get(i);
                try {
                    DateCuTime =
TempCuTime.parse(CurrentTime);
                } catch (ParseException e) {
                    e.printStackTrace();
                }
                LongCuTime = DateCuTime.getTime();
                HipLongTime.add(LongCuTime);

                // 加速度の大きさ計算と、必要あれば鉛直・
水平成分分解
                    TempX = (HipAccelX.get(i) -
HipAccelXAve);
                    TempY = (HipAccelY.get(i) -
HipAccelYAve);
                    TempZ = (HipAccelZ.get(i) -
HipAccelZAve);
                    TempValue2 =
Math.sqrt(Math.pow(TempX, 2) + Math.pow(TempY, 2) +
Math.pow(TempZ, 2));
                    if (AccelSeparate) {
                        TempValue = ((-1) * TempX *
HipAccelXAve + (-1) * TempY * HipAccelYAve + (-1) *
TempZ * HipAccelZAve) / HipAccelXYZ;
                        HipVirtAccelSize.add(TempValue /
256.0);
                        TempValue =
Math.sqrt(Math.pow(TempValue2, 2) - Math.pow(TempValue,
2));
                        HipHoriAccelSize.add(TempValue /
256.0);
                        HipAccelSize.add(TempValue2 / 256.0);
                    } else {
                        HipVirtAccelSize.add(0.0);
                        HipHoriAccelSize.add(0.0);
                        HipAccelSize.add(TempValue2 / 256.0);
                    }

            }

            // 足首センサ
            for (int i = 0; i < AnkleAccelX.size(); i++) {
                // データの時刻をLongに変換し格納
                CurrentTime = AnkleReceivedTime.get(i);
                try {
                    DateCuTime =
TempCuTime.parse(CurrentTime);
                } catch (ParseException e) {
                    e.printStackTrace();
                }
                LongCuTime = DateCuTime.getTime();
                AnkleLongTime.add(LongCuTime);

                    // 足首はセンサー姿勢変動が大きいた
め分割せずにサイズのみ計算
                    TempX = (AnkleAccelX.get(i) -
AnkleAccelXAve);
```

```
                    TempY = (AnkleAccelY.get(i) -
AnkleAccelYAve);
                    TempZ = (AnkleAccelZ.get(i) -
AnkleAccelZAve);
                    TempValue2 =
Math.sqrt(Math.pow(TempX, 2) + Math.pow(TempY, 2) +
Math.pow(TempZ, 2));
                    AnkleAccelSize.add(TempValue2 / 256.0);
                }


            // 加速度データ前処理
                // 移動平均をとる場合
                if (AccelMAcalc) {
                    // 腰センサ
                    for (int i = 0; i < AccelMAsize; i++) {
                        TempValue = 0.0;
                        for (int j = 0; j < (i + AccelMAsize + 1); j++) {
                            // 腰分割ありの場合は腰の水平成分を使
用
                            if (AccelSeparate) {
                                TempValue = TempValue +
HipHoriAccelSize.get(j);
                            } else {
                                TempValue = TempValue +
HipAccelSize.get(j);
                            }
                        }
                        TempValue = TempValue / (i + AccelMAsize
+ 1);
                        HipAccelSizeNC.add(TempValue);
                    }
                    for (int i = AccelMAsize; i < HipDataSize -
AccelMAsize; i++) {
                        TempValue = 0.0;
                        for (int j = i - AccelMAsize; j < (i +
AccelMAsize + 1); j++) {
                            // 腰分割ありの場合は腰の水平成分を使
用
                            if (AccelSeparate) {
                                TempValue = TempValue +
HipHoriAccelSize.get(j);
                            } else {
                                TempValue = TempValue +
HipAccelSize.get(j);
                            }
                        }
                        TempValue = TempValue / (2 * AccelMAsize
+ 1);
                        HipAccelSizeNC.add(TempValue);
                    }
                    for (int i = HipDataSize - AccelMAsize; i <
HipDataSize; i++) {
                        TempValue = 0.0;
                        for (int j = i - AccelMAsize; j < HipDataSize;
j++) {
                            // 腰分割ありの場合は腰の水平成分を使
用
                            if (AccelSeparate) {
                                TempValue = TempValue +
HipHoriAccelSize.get(j);
                            } else {
                                TempValue = TempValue +
HipAccelSize.get(j);
                            }
                        }
                        TempValue = TempValue / (HipDataSize - i +
AccelMAsize);
                        HipAccelSizeNC.add(TempValue);
                    }

                    // 足首センサ
                    for (int i = 0; i < AccelMAsize; i++) {
                        TempValue = 0.0;
                        for (int j = 0; j < (i + AccelMAsize + 1); j++) {
                            TempValue = TempValue +
AnkleAccelSize.get(j);
                        }
                        TempValue = TempValue / (i + AccelMAsize
+ 1);
                        AnkleAccelSizeNC.add(TempValue);
                    }
                    for (int i = AccelMAsize; i < AnkleDataSize -
AccelMAsize; i++) {
                        TempValue = 0.0;
                        for (int j = i - AccelMAsize; j < (i +
AccelMAsize + 1); j++) {
                            TempValue = TempValue +
AnkleAccelSize.get(j);
                        }
                        TempValue = TempValue / (2 * AccelMAsize
+ 1);
                        AnkleAccelSizeNC.add(TempValue);
                    }
                    for (int i = AnkleDataSize - AccelMAsize; i <
AnkleDataSize; i++) {
                        TempValue = 0.0;
                        for (int j = i - AccelMAsize; j <
AnkleDataSize; j++) {
                            TempValue = TempValue +
AnkleAccelSize.get(j);
                        }
                        TempValue = TempValue / (AnkleDataSize - i
+ AccelMAsize);
                        AnkleAccelSizeNC.add(TempValue);
                    }
                }

                // SGFilterを使う場合の計算
                if (AccelSGcalc) {
                    // Marcin Rzeźnicki氏の公開されている
SGFilter.javaの使い方を踏襲
                    // 腰センサ
                    double HipCoeffs[] = new double[HipDataSize];
                    double Hipdata[] = new double[HipDataSize];
                    double Hipresult[] = new double[HipDataSize];

                    for (int i = 0; i < HipDataSize; i++) {
                        if (AccelSeparate) {
                            Hipdata[i] = HipHoriAccelSize.get(i);
```

```
                } else {
                    Hipdata[i] = HipAccelSize.get(i);
                }
            }

            HipCoeffs =
SGFilter.computeSGCoefficients(AccelPartSize, AccelPartSize,
AccelDegree);
            ContinuousPadder padder1 = new
ContinuousPadder();
            SGFilter sgFilter1 = new SGFilter(AccelPartSize,
AccelPartSize);
            sgFilter1.appendPreprocessor(padder1);
            Hipresult = sgFilter1.smooth(Hipdata,
HipCoeffs);

            for (int i = 0; i < HipDataSize; i++) {
                HipAccelSizeNC.add(Hipresult[i]);
            }

            // 足首センサ
            double AnkleCoeffs[] = new
double[AnkleDataSize];
            double Ankledata[] = new
double[AnkleDataSize];
            double Ankleresult[] = new
double[AnkleDataSize];

            for (int i = 0; i < AnkleDataSize; i++) {
                Ankledata[i] = AnkleAccelSize.get(i);
            }

            AnkleCoeffs =
SGFilter.computeSGCoefficients(AccelPartSize, AccelPartSize,
AccelDegree);
            ContinuousPadder padder2 = new
ContinuousPadder();
            SGFilter sgFilter2 = new SGFilter(AccelPartSize,
AccelPartSize);
            sgFilter2.appendPreprocessor(padder2);
            Ankleresult = sgFilter2.smooth(Ankledata,
AnkleCoeffs);

            for (int i = 0; i < AnkleDataSize; i++) {
                AnkleAccelSizeNC.add(Ankleresult[i]);
            }
        }

        // MAもSGも使わない場合の計算
        if (AccelMAcalc || AccelSGcalc) {

        } else {
            for (int i = 0; i < HipDataSize; i++) {
                if (AccelSeparate) {

HipAccelSizeNC.add(HipHoriAccelSize.get(i));
                } else {
                    HipAccelSizeNC.add(HipAccelSize.get(i));
                }
            }

            for (int i = 0; i < AnkleDataSize; i++) {

AnkleAccelSizeNC.add(AnkleAccelSize.get(i));
                }
            }


        // 躍度計算
            // 腰センサ
            HipJerk.add(0.0);
            HipJerkAbs.add(0.0);
            for (int i = 1; i < HipDataSize; i++) {
                // 変化量を計算
                DefTime = (double)(HipLongTime.get(i) -
HipLongTime.get(i-1)) / 1000;
                DefAccel = HipAccelSizeNC.get(i) -
HipAccelSizeNC.get(i - 1);

                // 躍度計算
                HipJerk.add(DefAccel / DefTime);
                HipJerkAbs.add(Math.abs(DefAccel / DefTime));

            }

            // 足首センサ
            AnkleJerk.add(0.0);
            AnkleJerkAbs.add(0.0);
            for (int i = 1; i < AnkleDataSize; i++) {
                // 変化量を計算
                DefTime = (double)(AnkleLongTime.get(i) -
AnkleLongTime.get(i-1)) / 1000;
                DefAccel = AnkleAccelSizeNC.get(i) -
AnkleAccelSizeNC.get(i - 1);

                // 躍度計算
                AnkleJerk.add(DefAccel / DefTime);
                AnkleJerkAbs.add(Math.abs(DefAccel /
DefTime));

            }


        // 躍度判定
            // 腰センサ
            for (int i = 0; i < HipDataSize; i++) {
                if (HipJerkAbs.get(HipJerkAbs.size() - 1) >
HipJerkThreshold) {
                    HipAccelJudge.add(1.0);
                } else {
                    HipAccelJudge.add(0.0);
                }
            }

            // 足首センサ
            for (int i = 0; i < AnkleDataSize; i++) {
                if (AnkleJerkAbs.get(AnkleJerkAbs.size() - 1) >
AnkleJerkThreshold) {
                    AnkleAccelJudge.add(1.0);
                } else {
```

```
                AnkleAccelJudge.add(0.0);
            }

        }

        // 腰センサと足首センサのデータ合体
        Count = 0;
        Count2 = 0;
        TempValue = 0;

        for (int i = 0; i < (HipDataSize + AnkleDataSize);
i++) {
            if (Count >= HipDataSize) {

AccelObjectID.add(AnkleObjectID.get(Count2));

AccelDetected.add(AnkleDetected.get(Count2));

AccelReceivedTime.add(AnkleReceivedTime.get(Count2));
                AccelMacAd.add(AnkleMacAd.get(Count2));
                AccelRSSI.add(AnkleRSSI.get(Count2));
                AccelUUID.add(AnkleUUID.get(Count2));

AccelMajorID.add(AnkleMajorID.get(Count2));

AccelMinorID.add(AnkleMinorID.get(Count2));

AccelMeasuredPower.add(AnkleMeasuredPower.get(Count2))
;

AccelTemperature.add(AnkleTemperature.get(Count2));
                AccelVoltage.add(AnkleVoltage.get(Count2));
                AccelAccelX.add(AnkleAccelX.get(Count2));
                AccelAccelY.add(AnkleAccelY.get(Count2));
                AccelAccelZ.add(AnkleAccelZ.get(Count2));
                AccelVirtAccelSize.add(-999.0);
                AccelHoriAccelSize.add(-999.0);

AccelAccelSize.add(AnkleAccelSize.get(Count2));

AccelAccelSizeNC.add(AnkleAccelSizeNC.get(Count2));
                AccelJerk.add(AnkleJerk.get(Count2));
                AccelJerkAbs.add(AnkleJerkAbs.get(Count2));

AccelAccelJudge.add(AnkleAccelJudge.get(Count2));

AccelLongTime.add(AnkleLongTime.get(Count2));

                if (AnkleLongTime.get(Count2) <
LongLaCyStTime) {
                    AccelCluster.add(0);
                } else if (AnkleLongTime.get(Count2) >
LongLaCyFiTime) {
                    AccelCluster.add(2);
                } else {
                    AccelCluster.add(1);
                }

                TempValue = AnkleLongTime.get(Count2);

                if (Count2 < AnkleDataSize) {
                    Count2 = Count2 + 1;
                }

            }else if (Count2 >= AnkleDataSize) {
                AccelObjectID.add(HipObjectID.get(Count));
                AccelDetected.add(HipDetected.get(Count));

AccelReceivedTime.add(HipReceivedTime.get(Count));
                AccelMacAd.add(HipMacAd.get(Count));
                AccelRSSI.add(HipRSSI.get(Count));
                AccelUUID.add(HipUUID.get(Count));
                AccelMajorID.add(HipMajorID.get(Count));
                AccelMinorID.add(HipMinorID.get(Count));

AccelMeasuredPower.add(HipMeasuredPower.get(Count));

AccelTemperature.add(HipTemperature.get(Count));
                AccelVoltage.add(HipVoltage.get(Count));
                AccelAccelX.add(HipAccelX.get(Count));
                AccelAccelY.add(HipAccelY.get(Count));
                AccelAccelZ.add(HipAccelZ.get(Count));

AccelVirtAccelSize.add(HipVirtAccelSize.get(Count));

AccelHoriAccelSize.add(HipHoriAccelSize.get(Count));
                AccelAccelSize.add(HipAccelSize.get(Count));

AccelAccelSizeNC.add(HipAccelSizeNC.get(Count));
                AccelJerk.add(HipJerk.get(Count));
                AccelJerkAbs.add(HipJerkAbs.get(Count));

AccelAccelJudge.add(HipAccelJudge.get(Count));

AccelLongTime.add(HipLongTime.get(Count));

                if (HipLongTime.get(Count) <
LongLaCyStTime) {
                    AccelCluster.add(0);
                } else if (HipLongTime.get(Count) >
LongLaCyFiTime) {
                    AccelCluster.add(2);
                } else {
                    AccelCluster.add(1);
                }

                TempValue = HipLongTime.get(Count);

                if (Count < HipDataSize) {
                    Count = Count + 1;
                }

            }else {
                if ((HipLongTime.get(Count) - TempValue) <=
(AnkleLongTime.get(Count2) - TempValue)) {

AccelObjectID.add(HipObjectID.get(Count));
                    AccelDetected.add(HipDetected.get(Count));
```

```
AccelReceivedTime.add(HipReceivedTime.get(Count));
                    AccelMacAd.add(HipMacAd.get(Count));
                    AccelRSSI.add(HipRSSI.get(Count));
                    AccelUUID.add(HipUUID.get(Count));
                    AccelMajorID.add(HipMajorID.get(Count));
                    AccelMinorID.add(HipMinorID.get(Count));

AccelMeasuredPower.add(HipMeasuredPower.get(Count));

AccelTemperature.add(HipTemperature.get(Count));
                    AccelVoltage.add(HipVoltage.get(Count));
                    AccelAccelX.add(HipAccelX.get(Count));
                    AccelAccelY.add(HipAccelY.get(Count));
                    AccelAccelZ.add(HipAccelZ.get(Count));

AccelVirtAccelSize.add(HipVirtAccelSize.get(Count));

AccelHoriAccelSize.add(HipHoriAccelSize.get(Count));

AccelAccelSize.add(HipAccelSize.get(Count));

AccelAccelSizeNC.add(HipAccelSizeNC.get(Count));
                    AccelJerk.add(HipJerk.get(Count));
                    AccelJerkAbs.add(HipJerkAbs.get(Count));

AccelAccelJudge.add(HipAccelJudge.get(Count));

AccelLongTime.add(HipLongTime.get(Count));

                    if (HipLongTime.get(Count) <
LongLaCyStTime) {
                       AccelCluster.add(0);
                    } else if (HipLongTime.get(Count) >
LongLaCyFiTime) {
                       AccelCluster.add(2);
                    } else {
                       AccelCluster.add(1);
                    }

                    TempValue = HipLongTime.get(Count);

                    if (Count < HipDataSize) {
                       Count = Count + 1;
                    }

                 }else {

AccelObjectID.add(AnkleObjectID.get(Count2));

AccelDetected.add(AnkleDetected.get(Count2));

AccelReceivedTime.add(AnkleReceivedTime.get(Count2));

AccelMacAd.add(AnkleMacAd.get(Count2));
                    AccelRSSI.add(AnkleRSSI.get(Count2));
                    AccelUUID.add(AnkleUUID.get(Count2));

AccelMajorID.add(AnkleMajorID.get(Count2));

AccelMinorID.add(AnkleMinorID.get(Count2));

AccelMeasuredPower.add(AnkleMeasuredPower.get(Count2))
;

AccelTemperature.add(AnkleTemperature.get(Count2));

AccelVoltage.add(AnkleVoltage.get(Count2));

AccelAccelX.add(AnkleAccelX.get(Count2));

AccelAccelY.add(AnkleAccelY.get(Count2));

AccelAccelZ.add(AnkleAccelZ.get(Count2));
                    AccelVirtAccelSize.add(-999.0);
                    AccelHoriAccelSize.add(-999.0);

AccelAccelSize.add(AnkleAccelSize.get(Count2));

AccelAccelSizeNC.add(AnkleAccelSizeNC.get(Count2));
                    AccelJerk.add(AnkleJerk.get(Count2));

AccelJerkAbs.add(AnkleJerkAbs.get(Count2));

AccelAccelJudge.add(AnkleAccelJudge.get(Count2));

AccelLongTime.add(AnkleLongTime.get(Count2));

                    if (AnkleLongTime.get(Count2) <
LongLaCyStTime) {
                       AccelCluster.add(0);
                    } else if (AnkleLongTime.get(Count2) >
LongLaCyFiTime) {
                       AccelCluster.add(2);
                    } else {
                       AccelCluster.add(1);
                    }

                    TempValue = AnkleLongTime.get(Count2);

                    if (Count2 < AnkleDataSize) {
                       Count2 = Count2 + 1;
                    }
                 }
              }
           }


        // 各製造個数Cluster内での移動回数をカウント
           int TempAccelNum = 0;
           int MoveFlag = 0;
           long MoveStartTime = 0;
           long MoveFinishTime = 0;
           LongCuTime = 0;
           for (int i = 0; i < (HipDataSize + AnkleDataSize);
i++) {
              if (AccelCluster.get(i) == 1) {
                 if (MoveFlag == 0 && AccelAccelJudge.get(i)
== 1) {
                    MoveFlag = 1;
                    TempAccelNum = TempAccelNum + 1;
```

```
                    MoveStartTime = AccelLongTime.get(i);
                } else if (MoveFlag == 1 &&
AccelAccelJudge.get(i) == 0) {
                    MoveFlag = 0;
                    MoveFinishTime = AccelLongTime.get(i -
1);
                    TempValue = (double)(MoveFinishTime -
MoveStartTime) / 1000;
                    TempAccelMoveTime.add(TempValue);
                } else {
                    LongCuTime = AccelLongTime.get(i);
                }
            }
        }
        if (MoveFlag == 1) {
            MoveFinishTime = LongCuTime;
            TempValue = (double)(MoveFinishTime -
MoveStartTime) / 1000;
            TempAccelMoveTime.add(TempValue);
        }
        AccelMoveNum.add(TempAccelNum);


    // 各製造個数Cluster内での移動時間・停止時間を
集計
        TempValue = 0.0;
        for (int i = 0; i < TempAccelMoveTime.size(); i++)
{
            TempValue = TempValue +
TempAccelMoveTime.get(i);
        }
        AccelMoveTime.add(TempValue);

AccelStopTime.add(AccelCycleTime.get(AccelCycleTime.size(
) - 1) - TempValue);


    // GUI更新
        LastAccelResult_MoveNum =
AccelMoveNum.get(AccelMoveNum.size() - 1);

this.LastAccelResult_MoveNum_L.setText(String.valueOf(La
stAccelResult_MoveNum));
        LastAccelResult_MoveTime =
AccelMoveTime.get(AccelMoveTime.size() - 1);

this.LastAccelResult_MoveTime_L.setText(String.valueOf(La
stAccelResult_MoveTime));
        LastAccelResult_StopTime =
AccelStopTime.get(AccelStopTime.size() - 1);

this.LastAccelResult_StopTime_L.setText(String.valueOf(Las
tAccelResult_StopTime));
        repaint();


    // RWBASotherprocessスレッド起動用のフラグ更
新
        AccelFinishFlag = 1;
            } else {
                Thread.sleep(1000);
            }
        }

        return 0;
    }

}
```

ーーーーーーーーーーーーーーーーーーーークラスここまで.

```
/*
 * RWBASのデータ応用クラス
 * Masaki Kitazawa.
 *
 * RWBASの計測結果を用いて、最適化やシミュレーションへ応用するためのクラス
 */

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.concurrent.Callable;
import javax.swing.JPanel;


public class RWBASotherprocess extends JPanel
implements Callable<Integer> {
   private static final long serialVersionUID = 1L;

// 変数宣言
   static String OtherResultFile;
   String AddFileName = ".csv";


// スレッド構築
   public RWBASotherprocess(String ResultFile) {
      // スレッド構築時の初期処理を設定

      // ここではAgent-based simulation用の結果ファイル出力のみ
      // 結果ファイル名をサイクルタイム用に補正
      RWBASotherprocess.OtherResultFile = ResultFile + AddFileName;
   }


// メイン処理
   @Override
   public Integer call() throws Exception {
      while (RWBASmain.ReadingFlag == 1) {
         if (RWBASaccelanalyze.AccelFinishFlag == 1) {
            // RWBASaccelanalyzeの処理終了後に起動
            // 最適化やシミュレーションなどを設定可能

            // ここではAgent-based simulation用の結果ファイル出力のみ
            try {
               FileWriter fw0 = new FileWriter(OtherResultFile, false);
               PrintWriter pw0 = new PrintWriter(new BufferedWriter(fw0));

               int DataSize = RWBASaccelanalyze.AccelCycleTime.size();

pw0.println("ProductNumber,CycleStartTime,CycleFinishTime,"
                     +
"CycleTime,StopTime,MoveTime,MovementNumber");
               for (int i = 0; i < DataSize; i++) {
                  pw0.print(i);
                  pw0.print(",");

pw0.print(RWBASrssianalyze.CycleStartTime.get(i));
                  pw0.print(",");

pw0.print(RWBASrssianalyze.CycleFinishTime.get(i));
                  pw0.print(",");

pw0.print(RWBASaccelanalyze.AccelCycleTime.get(i));
                  pw0.print(",");

pw0.print(RWBASaccelanalyze.AccelStopTime.get(i));
                  pw0.print(",");

pw0.print(RWBASaccelanalyze.AccelMoveTime.get(i));
                  pw0.print(",");

pw0.print(RWBASaccelanalyze.AccelMoveNum.get(i));
                  pw0.println("");
               }

               pw0.close();

            } catch (IOException ex) {
               ex.printStackTrace();
            }


            // フラグを戻して処理終了
            RWBASaccelanalyze.AccelFinishFlag = 0;

         } else {
            Thread.sleep(1000);
         }
      }

      return 0;
   }

}
```

ーーーーーーーーーーーーーーーーーーーークラスここまで.