マルチモーダルセンサを用いたリアルタイム ADL認識・学習フレームワークの提案

大石伸之[1]　永間慎太郎[1]　沼尾雅之[1]

概要：センサデータを活用した ADL(日常生活動作) 認識技術の活用に注目が集まっている．これまで，RFID または加速度センサなど単一のセンサを用いて人の状態を検出する方法は多く提案されてきた．そして近年，より複雑かつ高精度な状態認識を実現するために，複数のセンサから得られるモダリティ情報を統合して用いるマルチモーダル ADL 認識手法が主流となりつつある．しかしながら，種類やサンプリングレートの異なる複数のセンサデータをリアルタイムで適切に統合する方法や，学習していない未知クラスへの対応という点においてはまだ課題が残されている．本稿では，マルチモーダルセンサを用いた ADL 認識・行動認識モデルの更新をリアルタイムで実現するためのフレームワークを提案する．フレームワークの実装は，オープンソースのログ収集ソフトウェアである Fluentd を独自プラグインで拡張したものを用いる．本フレームワークの応用例として，フレームワークを用いて開発した知的見守りロボット「見守りふくろう」の開発事例を示し，同見守りロボットを用いたの提案フレームワーク評価を行った．結果より，その動作の記述方法には改善の余地が残されているが，リアルタイムでの複数センサ統合という点では柔軟な処理を実現可能であることが示された．

Proposal of a Framework for Multimodal sensor-based ADL Recognition and Learning

NOBUYUKI OISHI[1]　SHINTARO NAGAMA[1]　MASAYUKI NUMAO[1]

1. はじめに

IoT・AI 技術を組み合わせて，異種複数センサからの得られる大量のセンサデータを収集・解析し現実社会へとフィードバックすることで流通，医療・介護，流通，防災など様々な社会的課題を解決しようとする試みが世界中で注目を集めている．センサデータを活用した ADL(日常生活動作) 認識もこうした取り組みの一環であり，特に少子高齢化が進む日本では，廃用症候群の防止やリハビリ進展度の判断等，高齢者の Quality of Life (QOL) の向上をサポートしていくため技術として特に重要な分野の一つでもある．

以前から，RFID または加速度センサなど単一のセンサを用いて人の状態を検出する方法は多く提案されてきた [2][7][9]．そして近年，より複雑かつ高精度な状態認識を実現するために，複数のセンサから得られるモダリティ情報 (画像音声等) を統合して用いるマルチモーダル ADL 認識手法 [3][11][12] が主流となりつつある．しかしながら，種類やサンプリングレートの異なる複数のセンサデータをリアルタイムで適切に統合する方法についてはまだ課題が残されているというのが現状である．センサ統合アプリケーション開発においては，設置したセンサごとにデータを取得・処理するためのプログラムが必要であり，さらにそれらの取得したデータを連携させるための記述や外部サービスとの連携のための設定が必要といったことから，管理するプログラムやファイルの数が多くなり，そして記述も煩雑になりがちであるという問題点が挙げられる．また，実際に環境に配置されたセンサは様々な外的・物理的要因から故障することも少なくなく，特に，リアルタイム性を求められるセンサ統合システムの場合，一つのセンサが何かしらの原因によって故障してしまった場合にも，正

[1] 電気通信大学大学院 情報理工学研究科
情報・ネットワーク工学専攻
The University of Electro-Communications

常に動作する部分には影響を及ぼさずに動作し続けることができるフェールソフトなシステムが求められる．

また一般に，機械学習による知識獲得には一般的に大量のデータが必要とされるが，センサデータは環境により数・種類・配置・サンプリングレート等が変化するというその性質上，ある統一されたフォーマットのデータを大量に手に入れることは難しく，またある環境において学習させた行動認識モデルが，他の環境において同様に機能することは基本的に期待できない．Ravi ら [9] によれば，加速度センサを用いた行動認識において，ある被験者から取得したデータを学習させ，同日そのモデルを同一被験者に対して適応した場合と別日に異なる被験者に対して適応した場合で，認識精度は 98%から 65%にまで落ちたとの結果もある．このように，あるケース (場所・人・目的) のために作成したモデルが，他のケースでもそのまま使用できるということは考えにくく，実際には，設置する環境の変化に合わせて認識モデルも随時更新していく必要がある [5]．

本稿ではこのような問題に対し，マルチモーダルセンサを用いた ADL 認識・行動認識モデルの更新をリアルタイムで実現するためのフレームワークを提案し，本フレームワークを用いて開発した知的見守りロボット「見守りふくろう」の開発事例を用いて，本フレームワークの評価を行う．

2. ADL 認識・学習フレームワーク

ここでは，提案するアクティブ・オンライン ADL 認識フレームワークの全体像 (アーキテクチャ) について述べた後，フレームワークの実装面や，実際にどのようにしてシステムを構築していくかの説明する．以下，図 1 は，提案するフレームワークの全体像を表したものである．

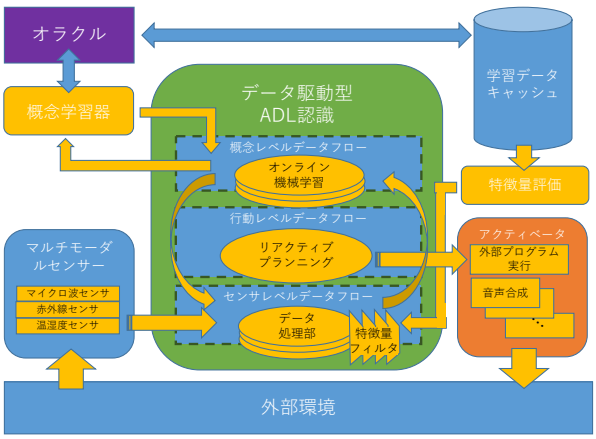

図 1　ADL 認識・学習フレームワーク全体像

2.1 データ駆動型 ADL 認識部

データ駆動型 ADL 認識部 (図 1 中央の緑色部分) が本フレームワークの核となっており，このデータ駆動型 ADL 認識部が外部環境に設置されている様々なセンサから取得されたデータを受け取り，センサから取得した外部の状況や認識した結果に応じて，アクティベータを通じて外部環境へと働きかける．このデータ駆動型 ADL 認識部の中では，データフローが論理的にセンサレベルデータフロー，行動レベルデータフロー，概念レベルデータフローの三種類に区別されており，それぞれのデータフローがそれぞれの役割を持っている．

2.1.1 センサレベルデータフロー

センサレベルデータフローには，様々なセンサから得られたデータがそれぞれのタイミングで流れてくる．それらのデータに対して，予め決められたルールに沿って必要なデータを抜き出し，整形・加工や複数データのマージといった処理を加え，センサから得られる生データを扱い易い形にするのがセンサレベルデータフローの役割である．センサレベルデータフローで加工されたデータを，その性質に応じて行動レベルデータフロー，概念レベルデータフローへ振り分ける．基本的に，行動レベルデータフローには，何かしらのアクションを認識したと判断されるレベルまで落とし込まれたデータが，概念レベルデータフローには，機械学習で用いるための特徴量として加工されたデータが送られる．図 2 は，例として非接触温度センサから得られた体温データと，マイクロ波ドップラーセンサから得られた心拍数・呼吸数データを組み合わせて，1 組のバイタルデータとしてマージする処理を表している．更に，得られたデータを元に「状態: 正常」というデータを付け加えた上で行動レベルデータフローへ送っている．

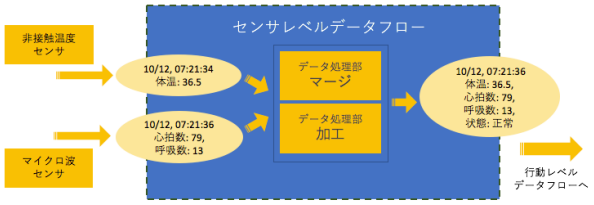

図 2　センサレベルデータフロー 処理例

2.1.2 行動レベルデータフロー

行動レベルデータフローには，センサレベルデータフロー内で行動レベルにまで加工されたデータ，または概念レベルデータフロー内で認識された行動認識結果が流れてくる．これらのデータを，リアクティブプラニングエンジンが処理し，見守りロボットアクティベータに命令を投げかける．リアクティブプランニングは，不確実な状況におけるプランニングの問題に対するアプローチの一種であり，現在のコンテキストに基づいてあらゆる瞬間においてただ 1 つの次のアクションを計算するタイムリーな方法で動作する自律エージェントによる行動選択のための手法である [8][10]．本研究も同様の手法を活用し，それぞれの瞬間において認識される結果にもとづいて見守りロボットがリアルタイムに，また途切れなく次のアクションを決定

できるようにする．行動レベルデータフローは，センサレベルデータフローや，概念レベルデータフローで認識された結果を，どのように活用するかを決定するデータ駆動型 ADL 認識部の中核とも言える場所であり，得られるデータ次第で様々なシステムの挙動を指定することができる．

2.1.3 概念レベルデータフロー

概念レベルデータフローには，センサレベルデータフローで加工されたマルチモーダル特徴量が流れてくる．これにオンライン機械学習を適応し，行動認識を行う．そして，認識された結果た行動レベルデータフローに送る．センサレベルデータフロー層では，主にセンサから送られてくるデータに直接 if-then ルールを適用することでデータを抽象化するというような比較的低次な操作を担当するのに対し，概念レベルデータフローは機械学習技術を用いた比較的高次なデータの抽象化を担当するという点でその役割が異なる．また，概念レベルデータフローでは，未知概念学習器・オラクルと連携して，未知概念の獲得も行う．これは既存の概念では説明できないデータ (特徴量) が流れてきた場合に，そのパターンに対応するラベルをオラクルに要求し，得られた正解ラベルを用いてオンライン機械学習で用いる行動モデルを更新するというものである．これにより，システムは変化する環境 (場所・人・目的) に適応するように，リアルタイムで認識モデルを更新することができる [6]．図 3 にそのプロセスの概要を示す．

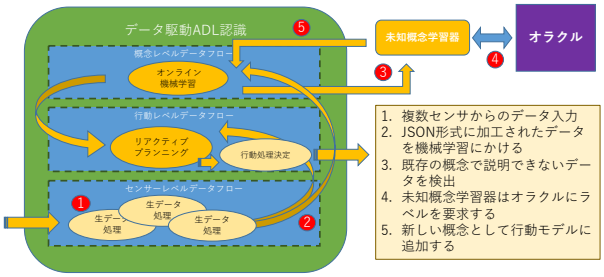

図 3 未知概念獲得の流れ

3. 提案フレームワークの実装

本章では，提案するフレームワークの実装面について述べる．本フレームワークの中心となるデータ駆動型 ADL 認識部にはオープンソースのログ収集ソフトウェアである Fluentd を独自のプラグインで拡張したものを用いる．Fluentd はデータ駆動型の処理が記述可能であり，また複数のイベントを並列的に処理可能ということから，多種多様なセンサからの送られてくる大量のデータを処理する必要のある本フレームワークとの相性も良い．また，軽量であり少ないリソース上で動作させることができる[*1]ということ，使用者のニーズに応じてプラグインを組み合わせることで豊富な機能を盛り込むことができることなどから，単なるログ収集といった用途だけでなく，IoT アプリケーション開発への応用にも適していると考えられる．ここでは，まず Fluentd 内のイベント構造について言及した上で，それぞれのデータフローをどのように表現するのかについて述べる．

[*1] あるインスタンスは 30-40MB ほどのメモリで 1 コア・1 秒あたり 13000 件のイベントを処理することが可能?

3.1 Fluentd のイベント形式

Fluentd 内を流れるデータは，それぞれが [tag, time, record] の組からなる．tag はピリオド'.' で区切られた構造をしており (e.g. room.1.temp)，Fluentd 内部でのルーティングを行う為に用いられる．また，time は UNIX 時間フォーマット，record は JSON 形式で表現される．例えば，以下の例では，tag は sensor.merged_vitaldata, time は 2017-11-03 19:35:32 +900, record は {“body_temp”: 36.8, “heart”: 79, “breath”: 12} に対応している．

ソースコード 1 Fluentd イベント形式

```
2017-11-03 19:35:32 +0900 sensor.microwave:
{ "heart":79, "breath":12}
```

3.1.1 Fluentd 設定ファイルの記述方法

Fluentd の設定ファイルの記述には，source，match，filter，system，label，@include という 6 つのディレクティブが用いられる．ここでは，主要な 2 つのディレクティブ (source，match ディレクティブ) についてのみ説明を行う．

3.1.1.1 source ディレクティブ

source ディレクティブは input プラグインと組み合わせて使う．これにより input プラグインを用いたデータ入力を有効にすることができる．例えば，24224 番ポートから tcp ソケットで Fluentd イベントの入力を受ける場合には，ソースコード 2 のように in_forward プラグインを使えばよい．C++，Java, Python, Ruby，Go 等のメジャーな言語には fluent-logger という Fluentd インスタンスに Fluentd イベント形式データを送るためのインターフェースが用意されているので，これと in_forward プラグインを組み合わせれば，センサデータを Fluentd に送ることができる．input プラグインには他にも，テキストファイルの末尾からログを取り出す in_tail プラグイン，HTTP で待ち受ける in_http プラグインなどがある．

ソースコード 2 source ディレクティブ記述例

```
<source>
  @type forward
  port 24224
</source>
```

3.1.1.2 match ディレクティブ

match ディレクティブは output プラグインと組み合わせて使う．これにより output プラグインを有効化し，更

にイベント [tag, time, record] の tag のマッチングにより，ルーティング及びどのような処理を行うかを指定することができる．tag のマッチングには以下のようなワイルドカードを用いることもできる ('*', '**', '{}')．例えば，ソースコード 3 のように action.*にマッチするタグを持つイベントに out_file プラグインを使うことで，認識されたアクションを全てファイルに書き出すといった処理を記述することができる．

ソースコード 3　match ディレクティブ記述例

```
<match action.*>
  @type file
  path /var/log/fluent/vitaldata_all
</match>
```

3.1.2　Fluentd による 3 種のデータフローの区別

Fluentd の tag 名ベースのルーティングを使えば，データ駆動型 ADL 認識部内部の 3 つのデータフロー (センサレベルデータフロー，行動レベルデータフロー，概念レベルデータフロー) のデータフローのレベルを容易に区別することができる．上述の通り，tag はピリオド'.' で区切られた構造をしている．このピリオドで区切られた tag に対して，左から順番第 1 レベル，第 2 レベル…と名前を付けることにする．本フレームワークでは，この tag の第 1 レベルをデータフローのレベルを区別するために用いる．例えば，第 1 レベルが sensor から始まる場合は，そのデータはセンサレベルデータフローを流れていることを意味するし，第 1 レベルは action から始まる．また，センサレベルデータフローで加工したデータをアクションレベルデータフローに送りたいという場合は，tag の第 1 レベルを sensor から action に変えて送信すれば良い (図**??** 参照)．

コンセプトレベル データフロー	concept.**
アクションレベル データフロー	action.**
センサレベル データフロー	sensor.**

図 4　データフローごとの Fluentd イベント命名規則

3.2　独自プラグインによる Fluentd の機能拡張

Fluentd はそれ自体がシンプルかつ協力なソフトウェアだが，既存のプラグインだけでは本フレームワークが求めているような機能 (特に非同期的に送られてくる複数のデータを統合するという点) を実現することは難しい．そこで，Fluentd の機能を拡張するため以下の 4 つのプラグインを独自に作成した．

3.2.1　reocod_merger プラグイン

非同期で送られてくる複数のイベントをマージし，複数イベントにまたがる処理の記述を可能にするプラグイン．main_tag に指定したイベントを受け取った際，sub_tag に指定されているイベントの内容と結合して新たなイベントを定義して Fluentd 内に出力する．なお，sub_tag は複数指定することができる．イベントの結合タイミングはオプションで指定することが可能．

3.2.2　condition_checker プラグイン

センサ統合アプリケーションで多々必要となる，取得したセンサの値に応じた複数の条件分岐やレコードの整形・加工を簡潔に記述するすることができる．

3.2.3　active_online_learning プラグイン

Fluentd とオンライン学習プラットフォーム Jubatus を連携させ行動認識・未知概念学習を行う．オンライン機械学習向け分散処理フレームワークである Jubatus を用いたアクティブ・オンライン学習用のプラグインであり，既存の Jubatus プラグインを拡張して作られている．train と analyze の 2 つのモードが存在しており，train モードでは学習モデルの訓練を行う．analyze モードでは，Fluentd イベントを分類・推薦・異常検知等，Jubatus が提供しているプラグインを使用して分析した結果をリアルタイムで受け取ることができる．

3.2.4　action_serialize プラグイン

アクションのシリアライズを行う．Fluentd の file プラグインを使ったログ機能と Python ライブラリの watchdog のようなファイルの変更を監視し，ファイルが変更が加えられた際に何かしらの処理を行うことができるようなプログラムを組み合わせることで，Fluentd 内のデータを使う外部のプログラムをシーケンシャルに実行できる．Fluentd には，Fluentd から外部のプログラムを実行するための exec プラグインというものが標準で用意されているが，プログラムの実行タイミングを適切に制御できず前回の処理が終了していないにも関わらず次の処理が始まってしまうといった不都合が生じる場合がある．例えば，あるイベントが流れてきた際にロボットに発話をさせるという処理を記述したい場合，連続で発話イベントが流れてきた場合にプログラムが直列化されていないと音声が重なって出力されてしまう．

4.　アプリケーション: 見守りふくろう

図 5 は，株式会社ワイヤレスコミュニケーション研究所と共同で開発した知的見守りロボット「見守りふくろう」である．内部に Raspberry Pi3 を搭載しており，その上で本フレームワークを用いて開発したシステムが動作している．この見守りふくろうは独居老人や介護施設入居者の見守りやコミュニケーションの促進を用途として想定しており，顔認識カメラ，マイクロ波ドップラーセンサ，非接触温度センサ，温湿度センサ，マイク，スピーカーなどが取り付けられている．本体に搭載されているセンサ以外に

も，スマートフォンやマットレスセンサや RFID などからも Wi-Fi 経由でデータを入力することが可能．

図 5　マルチモーダルセンサを搭載した見守りフクロウ

現在，見守りふくろうには以下の機能が実装されている．

- 顔認識カメラを用いた個人の特定
- 顔認識カメラに写った人物の感情分析
- バイタルデータ (心拍数・呼吸数・体温) の測定・記録
- ADL(日常生活動作) 認識
- SADL(社会的日常動作)[*2]認識 [4]
- 簡単なあいさつ・会話
- 入退室管理
- 温湿度管理

5. 評価実験

株式会社ワイヤレスコミュニケーション研究所様，介護老人保健施設ゆい様協力の元に，知的見守りロボット「見守りふくろう」を施設内に設置してデータの収集を行った．見守りふくろうには顔認識カメラ，マイクロ波センサ，赤外線表面温度センサ，マイクロ波ドップラーセンサ，温湿度センサが搭載されている．この見守りふくろうを 2018 年 1 月 18 日の正午から 2018 年 1 月 19 日の 18 時までの 30 時間設置し，データの収集および動作確認を行った．本実験では，知的見守りロボットがセンサから得られるデータに応じて定義されたシナリオに沿って継続的に動作することができるかを確認する．また，構築したシステムが単位時間あたりにどれ程のデータ量を処理することができているのか，また，システムのリアルタイム性 (センサから値を得てからシナリオに沿ってアクションをするまでにかかる時間) についても検証する．

5.1 使用シナリオ

シナリオは以下のように定義した．DAG 形式で表したものを図 6 に示す．

(1) 顔認識カメラが顔を認識した時，登録済みの顔が 2 つ以上あった場合に見守りロボットは挨拶をする
(2) 顔認識カメラが顔を認識した時，登録済みの顔が 1 つだけ存在し，かつ顔が中央に位置していた場合，見守りロボットは挨拶をして，また同時にセンサから取得したバイタルデータ (体温・心拍数・呼吸数) をデータベースへ保存する
(3) 顔認識カメラが顔を認識した時，登録済みの顔が 1 つだけ存在し，かつ顔が中央になかった場合，バイタルデータを取得できるように，見守りロボットは対象の人が中央に来るよう呼びかける
(4) 顔認識カメラが顔を認識した時，登録済みの顔が 1 つも無く，また 5 人以上が笑顔の場合，見守りロボットは「なんだか楽しそうですね」と話しかける

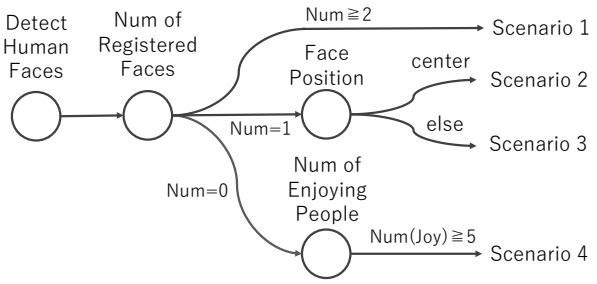

図 6　実験用シナリオ

6. 結果

6.1 処理データ量

見守りロボットを設置した 30 時間のうち，データフローの上を流れたデータの数は 277596 件で，ログファイルのサイズは 59.4MB となった．つまり，平均で毎秒 2.6 個ほどのデータがデータフロー上に流れていたことになる．1 秒あたりにデータフロー上を流れた最大のデータ数は 197 件だった．この中には，MQTT ブローカーとの接続切れた際のエラーメッセージを多く含んでいるが，その際にも通常のセンサデータは問題なく処理されている．

6.2 アクションのリアルタイム性

見守りロボットを設置した 30 時間のうち，図 6 に定義したシナリオの実行回数はそれぞれ以下のようになった．こ

表 1　シナリオ実行回数

	シナリオ 1	シナリオ 2	シナリオ 3	シナリオ 4
実行回数	1	97	40	1

の計 139 件全てのシナリオにおいて，本システムは顔認識カメラが顔を認識してから 1 秒以内にシナリオの最後まで実行することができていた．なお，本実験時はミリセカンドの精度でデータを処理していなかったため，これ以上の精度でのリアルタイム性を検証することはできていない．

6.3 発話イベントのシリアライズ

イベントのシリアライズに関しては，ログデータからの評価はできないが，現場で確認した限りでは重複なく話すことができており，正しく機能していることが確認できた．

[*2] Social Activities of Daily Living

6.4 Fluentd 設定ファイルの行数

のシナリオを定義するための Fluentd 設定ファイルの行数は 267 行．その内，<match></match>，<record></record> といったディレクティブの開始/終了記号のみから成る行数は 109 行，@type mqtt といった様にプラグインの種類を指定する行は 32 行，残りが具体的な処理の記述となっていた．

7. 考察

実験結果より，システムは概ね想定していた通りの動作をすることが確認できた．リアルタイム性については，より細かい精度での検証をするには，システムの設定をミリセカンドに対応していく必要がある．途中，MQTT ブローカーとの接続が切れてしまったことは予定外だったが，その間も MQTT でデータを飛ばす処理以外は正常に機能していた点は，Fluentd の並列的な処理が可能という長所が確かに活かされていると言える．

しかし，Fluentd を採用したことの長所も見えてきた一方で，短所も見られた．Fluentd 設定ファイルは，その記法の性質上，行数が膨らみやすい．今回，比較的単純なシナリオを 4 つ記述しただけで 267 行と，既に可読性が高いとは言えない状態である．また，実際に設定ファイルの中身を分析してみると，同様の記述が何箇所にも記述されているということにも気づく．例えば，MQTT ブローカーにデータをパブリッシュするための処理を記述する際には，送りたいデータに対応する全ての tag に対して 4 のような処理を記述する必要がある．しかし，10 行目のトピック名を指定する箇所以外は全て同様の記述となっており，無駄が多い．リアルタイムでの複数センサ統合という点では柔軟な処理を実現することができたが，その記述法に関してはまだ改善の余地があると考えられる．

ソースコード 4　MQTT プラグイン記述例

```
<match tag.to_mqtt_broker>
  @type mqtt
    host hostname
    port port_no
    <security>
      username 'username'
      password 'password'
    </security>
    topic_rewrite_pattern '^([\w\/]+)$'
    topic_rewrite_replacement 'topic_name'
    <format>
      @type json
      add_newline false
    </format>
</match>
```

8. おわりに

本稿では，マルチモーダルセンサを用いたリアルタイム ADL 認識・学習フレームワークを提案した．評価実験を行い，本フレームワークを用いて開発した見守りロボットが期待通り動作していることを確認できた．今後は，より詳細な Tag 命名規則や使用する基本プラグインの策定を行い，Fluentd の設定ファイルを生成するためのフレームワーク用言語を新たに定義することで，処理を記述する際の冗長さや可読性の悪さといった問題に対処していく．

謝辞

本研究は，JSPS 科研費 JP17H01823「無負荷センサ統合による見守りシステムの構築法」の助成を受けたものです．

参考文献

[1] Fluentd Project. What is Fluentd? https://www.fluentd.org/architecture.

[2] D. Fortin-Simard, J. S. Bilodeau, K. Bouchard, S. Gaboury, B. Bouchard, and A. Bouzouane. Exploiting passive rfid technology for activity recognition in smart homes. *IEEE Intelligent Systems*, Vol. 30, No. 4, pp. 7–15, July 2015.

[3] Vit Libal, Bhuvana Ramabhadran, Nadia Mana, Fabio Pianesi, Paul Chippendale, Oswald Lanz, and Gerasimos Potamianos. Multimodal classification of activities of daily living inside smart homes. In Sigeru Omatu, Miguel P. Rocha, Jos￥'eBravo, FlorentinoFern￥'andez, Emilio Corchado, Andr￥'esBustillo, Juan M. Corchado（編）, *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, pp. 687–694, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[4] Shintaro Nagama and Masayuki Numao. Iot-based emotion recognition robot to enhance sense of community in nursing home. In *2018 AAAI Spring Symposium Series: Beyond Machine Intelligence: Understanding Cognitive Bias and Humanity for Well-being AI*, 2018.

[5] Masayuki Numao and Shuya Masuda. Non-restrictive continuous health monitoring by integration of rfid and microwave sensor. In *2016 AAAI Spring Symposium Series: Well-Being Computing: AI Meets Health and Happiness Science*, 2016.

[6] Nobuyuki Oishi and Masayuki Numao. Active online learning architecture for multimodal sensor-based adl recognition. In *2018 AAAI Spring Symposium Series: Beyond Machine Intelligence: Understanding Cognitive Bias and Humanity for Well-being AI*, 2018.

[7] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2847–2854, June 2012.

[8] Collins G. Pryor L. Planning for contingencies: a decision-based approach. *Journal of Artificial Intelligence Research 4*, pp. 287–339, 1996.

[9] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *Proceedings of the 17th Conference*

on Innovative Applications of Artificial Intelligence - Volume 3, IAAI'05, pp. 1541–1546. AAAI Press, 2005.

[10] CTI Reviews. *Artificial Intelligence*. Cram101, 2016.

[11] N. Roy, A. Misra, and D. Cook. Infrastructure-assisted smartphone-based adl recognition in multi-inhabitant smart environments. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 38–46, March 2013.

[12] M. Stikic, T. Huynh, K. Van Laerhoven, and B. Schiele. Adl recognition based on the combination of rfid and accelerometer sensing. In *2008 Second International Conference on Pervasive Computing Technologies for Healthcare*, pp. 258–263, Jan 2008.