平成 26 年度　修士論文

センサ情報を用いた洗濯物状態判定に関する研究

龍谷大学　大学院　理工学研究科　情報メディア学専攻

T13M072　広瀬　大樹

指導教員　三好　力　教授

内容梗概

家事代行は，家事を代行することでユーザの家事に対する労力や作業時間を削減し，自由な時間を提供するものである．介護サービスにおける家事の定義は，料理・洗濯・掃除・買い物・ゴミ出し・衣類整理・衣類修繕，となっている．この中で，料理は電子レンジなどの電気調理器，洗濯は洗濯乾燥機，掃除はお掃除ロボット，買い物は宅配サービスなどの自動化・代行サービスが存在する．しかし，衣類整理にあたる洗濯物の片付けは未だ手作業が一般的である．これは，衣類が対象となるので形状が不定形で取り扱いが複雑になり，ロボットによる自動化が難しいためである．

そこで我々はカラー画像だけでなく深度情報も合わせて用いることで洗濯物を比較的容易に扱えるようになるのではないかと考えた．

本研究では洗濯物の状態を複数の状態に分けて定義し，ゲームコントローラ Kinect によるカメラ画像と 3 次元情報を用いて定義とのマッチングを行なう事により洗濯物の状態判定システムを開発する．

Abstract

In recent years, housework automation has become popular with the rise of robot technology. However, tidying laundry is still manual. Automation by machine is difficult, because clothing is an irregular complex shape. We thought it can handle the laundry by combining depth information and color image. The purpose of this study is to develop a laundry state determination system. This study define state of laundry by dividing into 4 states, and develop laundry state determination system using RGB image and 3D information. The results of experiment of the proposed method suggest that the system was possible to accurately determination the state of the laundry by using depth information and RGB camera image with Kinect.

目次

第 1 章 はじめに

1.1 研究背景

昨今，家庭用ロボットの普及に伴い，洗濯乾燥機や掃除ロボットなどの機械による，家事の自動化・代行サービスが一般的なものとなってきている．家事代行は，家事を代行することでユーザの家事に対する労力や作業時間を削減し，自由な時間を提供するものである．

本論文における「家事」とは，介護サービスにおける家事の定義より，掃除・料理・買い物・洗濯・衣類整理の 5 つとなっている[1][2]．介護サービスとはこれらをまとめて代行するものであり，また個別で自動化・代行を行うものも存在する．

掃除では iRobot 社の Roomba などの掃除ロボットが一般的なものとなってきている[3]．最近では床だけでなく窓拭きも可能なもの[4]も販売されており，ロボットによる掃除の自動化は進み続けている．料理における自動化・代行は電子レンジなどの電気調理器，またはデリバリーサービスによって可能となっている．また，食器洗浄機による食器の洗浄の自動化も存在する．買い物にはネットショッピングと宅配サービスによる代行サービスが存在する．洗濯においては洗濯乾燥機を用いての自動化が一般的なものとなっている．

しかし，衣類整理にあたる自動化・代行サービスは一般家庭用でないものが多く，自動化されていても工場用などの大規模なものを一般家庭に取り入れることは難しい．これは，衣類が対象となるので形状が不定形で取り扱いが複雑になり，ロボットによる自動化が難しいためであると考えられる．

表 1.1：一般家庭用自動化・代行サービスの例

掃除	料理	買い物	洗濯	衣類整理
掃除ロボット ハウスクリーニング	電気調理器 フードデリバリー	ネットショッピング	洗濯乾燥機 クリーニング	-

1.2 研究目的

現在，家庭用で洗濯物の整理をロボットなどで自動化するものは一般的でない．洗濯物の整理の自動化が実現すれば，長時間座ったままでの作業の低減や箪笥の開け閉め動作による腰の負担の軽減による高齢者など足腰の弱い人の負担軽減が望める．また，衣類整理の時間を他の作業へ割り当てることで多忙な家庭での時間節約も望むことができる．

本研究を行うにあたり，ロボットによる洗濯物整理の自動化に求められる処理を分析した結果，洗濯物の状態認識，洗濯物山からの取り出し処理，靴下などの小物のペアリング処理，洗濯物の片付け処理の 4 つに分割できることがわかった．洗濯物の状態認識が正確に行われていないとそれ以降の処理が確実に行われないという点から，洗濯物の状態認識は重要であると考えられる．よって，本研究では複数のセンサ情報を用いて洗濯物の状態を認識，その状態遷移命令を出力するシステムの開発を目的とする．センサ情報の取得は，RGB カメラと赤外線センサが搭載されていて比較的安価に手に入る Kinect[5]を用いた．

第 2 章 Kinect とは

Kinect は Microsoft から販売されている Xbox360 専用のゲームデバイスである．2010 年 11 月 4 日に米国で発売され，同年 11 月 20 日に日本でも発売された．Kinect はコントローラを使わずに操作ができる体感型のゲームシステムが特徴で，ジェスチャーや音声認識を用いた直感的なプレイが可能である．このようなユーザインタフェースは Natural User Interface(NUI)と呼ばれ，従来のデバイスは，ゲームを動かすために情報を入力するコントローラを使っていたのに対し，Kinect は内蔵されている 3 つのセンサを用いて高速に演算するプロセッサによって処理することで，人間の検出，プレイヤーの姿勢の認識，リアルタイムの動きの把握をし，ゲームの操作を可能としている．[6]内蔵されているセンサには，映像の出力や写真を撮るためにカラー画像を出力する RGB カメラ，Kinect から対象までの距離を測定できる近赤外線距離画像センサ，音声を認識するための 4 つのマイクセンサが搭載されている．近赤外線距離画像センサは近赤外光パターンを広範囲にレーザを照射する近赤外光プロジェクタとレーザ照射された近赤外光パターンを撮影し，撮影された画像から奥行きを計算する近赤外カメラ(3D 奥行きカメラ)で構成されている．

近赤外線距離センサでは三角測量を利用した計測法を採用している．まず，Kinect に搭載された近赤外レーザから対象物にスポット光をランダムに配列したようなパターンを複数投影し，対象物に映ったパターンをカメラでとらえる(図 2.1)．そして，とらえたパターンのカメラからの見込み核を求め，レーザから対象物にパターンを照射した角度と，レーザとカメラの距離を基に，対象物までの距離を算出する．パターン内のスポット光を区別できれば，1 回の投影によってカメラがとらえた画面内の物体の複数店の距離計測が可能となる．[6]

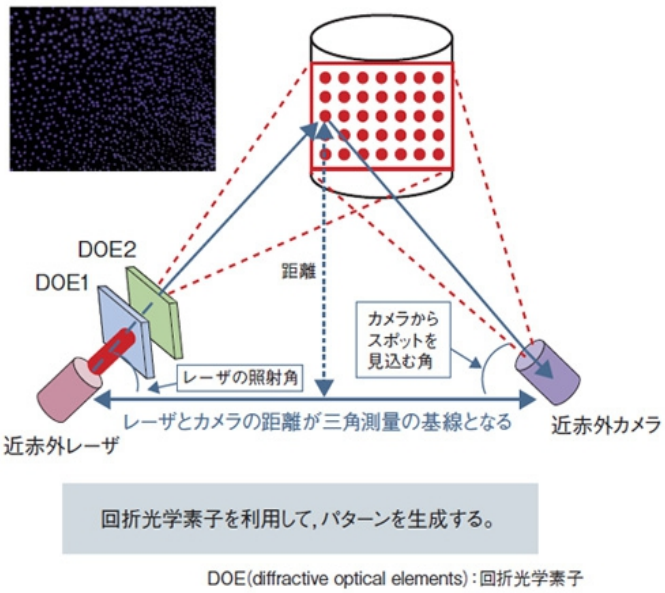


図 2.1：三角測量による対象物の距離測定

また，Kinect には USB インタフェースが付属されていることため，コンピュータにも接続可能であり，開発元の Microsoft は，非商用の場合に限ってコンピュータでも Kinect を利用することを認めており，「内部アルゴリズムにアクセスしないこと」と「Xbox 向けのゲームの不正利用に使用しないこと」を条件として開発を許可している．さらに，発売当初は Microsoft から公式なデバイスドライバの提供予定はなかったが，2011 年 6 月 16 日に windows7 での使用を可能にするソフトウェア開発キット「Kinect for Windows SDK(MS SDK)」のベータ版をリリースした．また，Kinect を Microsoft と共同開発した PrimeSense などが中心となって実装しているオープンソースライブラリの「OpenNI」によって Kinect を含めた NI デバイス全般を扱うことができる．ライセンスは GPL および LGPL である．開発環境としては，NI デバイスのドライバやインターフェースを担当するセンサ・モジュール，およびユーザ/スケルトントラッキングや各種ジェスチャーを認識できる「NITE」を OpenNI と併せてインストールするのが一般的である．MS SDK と OpenNI の仕様を表 2.1 に示す．

表 2.1：MS SDK,OpenNI の仕様[9]

項目	公式SDK	OpenNI
対応OS	Windows 7(x86,x64) Windows 8 Developper Preview	Windows XP, Vista, 7(それぞれx86,x64) Windows 8 Developper Preview Linux(Ubuntu)(x86,x64) Mac OS Android
開発言語	C++,C#	C,C++,C#,Java
対応デバイス	Kinect	Xtion Pro,Xtion Pro LIVE(公式),Kinect(非公式)
カメラ解像度	1280x1024,640x480	1280x1024,640x480
距離解像度	640x480	640x480
距離範囲	850mm-4,000mm	500mm-10,000mm
ユーザー認識解像度	320x240,80x60	640x480
スケルトントラッキングのためのポーズ	不要	OpenNI 1.4.0.2以降では不要
ユーザートラッキング数	7人	仕様なし
スケルトントラッキング数	2人	仕様なし
部分的なトラッキング	不可	可
ミラー機能	不可	可
カメラと距離のズレ補正機能	可	可
カメラ、距離の保存、再生	不可	可
ジェスチャー検出機能	不可	可
複数デバイスの操作	可	可

OpenNI には OS が Windows に限定されない，画像をミラー反転させたり，RGB 画像と深度画像のビューポイントを一致させる関数や，Kinect から取得したデータをファイルに書き出せるなどの利点がある．そのため，今後の研究を考慮して本研究では OpenNI を使って研究を進めた．

図 2.2：Kinect 外観

第 3 章 既存の家事代行ロボット

3.1 Roomba

Roomba とは，iRobot 社が製造・販売している家庭用のロボット掃除機である． 直径 34cm 程度の円盤状で，高さは 9cm 以内となっている．接触センサが組み込まれたバンパーや赤外センサが搭載されており，壁や家具との接触や段差を感知することで方向転換や転落を防いでいる．Roomba は掃除している部屋の地図を作成せず，「らせん移動」「壁伝い」「ランダムな角度に直進」など単純な動作で構成されている．このため Roomba と人間の掃除の仕方を比較すると，Roomba の方が時間がかかり，特定の箇所を何度も重複して掃除するのに別の場所は一回しか通らないとか全く通らないということも起きうる．

「Virtual Wall Unit」という仮想的な壁を設置することで進入禁止のエリアを指定することができる．機種によっては光センサによって汚れやゴミを感知し，そこを重点的に掃除する事が可能となっている．

図 3.1：Roomba

3.2 ホームアシスタントロボットによる掃除片付け

人が生活する環境には，人が使う様々な道具や装置が存在する．ホームアシスタントロボットはこれらを操作し家事・介護支援を行う．ここでは，家事・介護支援の基本作業のうち，トレイに乗せた食器の運搬，洗濯機への洗濯物投入，ほうきを使った床掃除を一台のロボットで連続して実現するための要素技術が開発されている．[7]

3.2.1 ホームアシスタントロボットの各機能

・環境認識機能

ホームアシスタントロボットで上記のような作業を行うには，家具や道具，洗濯物などを認識する能力が必要である．家具姿勢推定では，LRF(LaserRangeFinder)のスキャンデータと，ステレオカメラで撮影した画像データを併用し，イスのような模様の少ない物体に対しても高い姿勢推定精度で認識が可能になっている．衣類発見では，布が持つしわを画像特徴としてロボットに学習させることで，視覚を用いた認識が可能となっており，床やイスの上に衣類が無造作に置かれていても，これを認識して収集するような作業が実現された．

・行動生成機能

家具や道具などを操作するために，3 次元幾何モデルを規範とした行動生成システムが構築されている．ロボットの操作対象を三次元幾何モデルとして定義しておき，さらに操作点情報を与えてく．ロボットは前述の環境認識機能を用いて対象物を見つけ，そこへアプローチしながら，この操作情報を用いて自身の動作を生成する．このとき，ロボットと操作対象の間には認識や移動に伴う誤差が入り得るが，この誤差を解消できるよう，操作姿勢を再生成する機能が備えられている．

・失敗検知・動作やり直し機能

視覚・力覚などの外界センシング情報や，計画時の姿勢と実作業時の姿勢を比較するなどして，作業状態を監視する機能が搭載されている．例えば，布の掴み上げが成功したか，洗濯機の開閉ボタンを押すことはできたか，ゴミを取りやすい向きでほうきを把持できているか，などである．これらの作業で失敗した場合に，それを認識し，やり直し行動を生成してリトライすることで，失敗をリカバリして次々と作業をこなすことが可能となっている．

3.2.2 ホームアシスタントロボット実験機の構成

この研究で用いられたホームアシスタントロボットの実験機は以下のような構成となっている.

・様々な姿勢で物体に作用できる腰軸＋双腕アーム構成

・器用な物体操作を可能にする片手 6 自由度の多指ハンド

・空間内を広く移動するための台車機構

・ステレオカメラ, LRF(Laser Range Finder), 腕部力センサ等の外界センサ

(1)作業開始　(2)トレイ運搬　(3)トレイ運搬

(4)衣類発見・掴み上げ　(5)洗濯機ボタン押し　(6)ドア閉め

(7)イスどかし　(8)テーブル下掃除　(9)両手掃除

図 3.2:ホームアシスタントロボットの動作

3.3 GUI 操作によるロボットへの服の畳み方の教示

ロボットに関する知識を持たないユーザが，家庭用ロボットに対して自分好みの作業方法を簡単に教示するため，ロボットにさせたい作業をグラフィカルユーザインタフェース(GUI) 上で操作することで，ロボット自体を意識する必要のない直感的な教示を実現する手法である．[8]GUI には衣類のモデルが表示されており，ユーザは簡単なマウスドラッグ操作によって画面上のモデルを折り畳む．システムは操作結果に従い，ロボットの行動手順を生成し，ロボットは指示通りに実際の衣類を折り畳む．GUI には実世界における制約が考慮されているため，ロボットに可能な動きのみを効率よく教示することができる．

3.3.1 グラフィカルユーザインタフェース

ユーザは衣類を展開された状態で指定された実世界のステージの上に置く．続いて天井に取り付けられた web カメラによって服の形状をキャプチャし，GUI 上で 2 次元のモデル化された衣類を表示する(図 3.3)．ユーザは衣類のモデルの折り畳み方・手順をドラッグ＆ドロップ操作で入力する．ロボットが実世界で作業不可能な入力を行っている場合，視覚的に衣類の色を変化させることでロボットの制約を視覚的にユーザに提示する仕組みが組み込まれている(図 3.4)．システムは，操作結果に従いロボットの行動手順を生成し，実世界でロボットは指示通りに衣類を折り畳む．

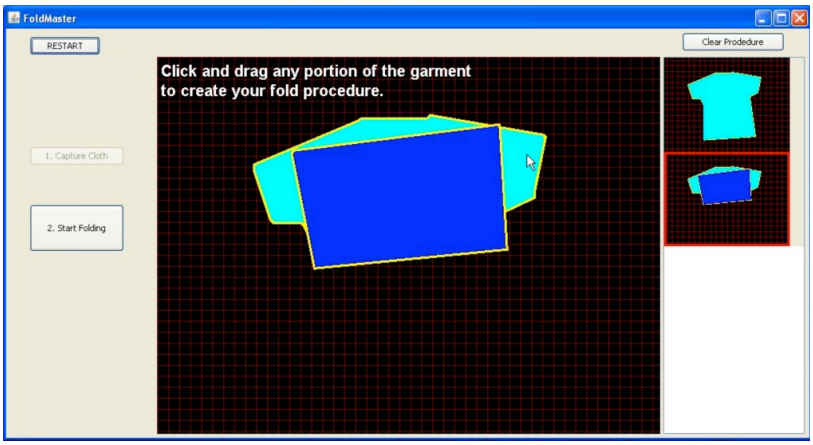


図 3.3：服畳み表示インタフェース

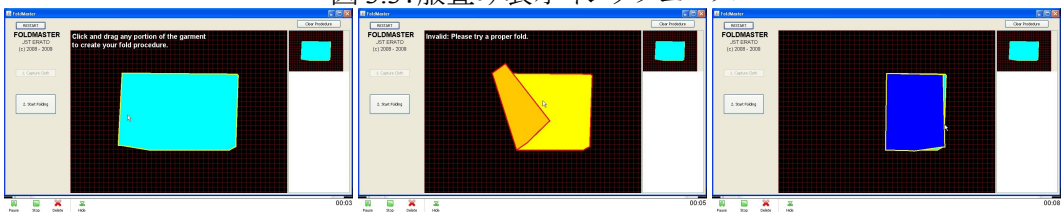


図 3.4：ロボットの物理的な制約を視覚的に表示する

3.3.2 服畳みロボット

服畳みロボットは，4 つの車輪，およびそれを駆動するモータ，服をつかむためのハンド，システムと無線通信するための Bluetooth，それらを制御するマイクロコントローラで構成されている．ロボットにはあらかじめ，車輪の「前進」「後進」「右旋回」「左旋回」や，ロボットハンドの「つかみ」「放し」の動作が予めマイクロコントローラ内に組み込まれている．コンピュータから各動作に対応する制御命令を送受信することで，ロボットの動作を制御することが可能である．ロボット上部には，システムが Web カメラでロボットの位置を認識するためのビジュアルマーカが取り付けられている．ロボットのアームの先端にはハンガーのような手先を取り付けることで，ズボンや長袖など長い辺の衣服を畳むことが可能である．

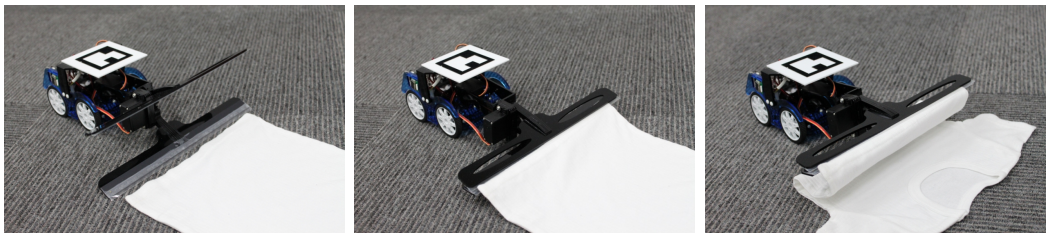


図 3.5：衣類の把握動作

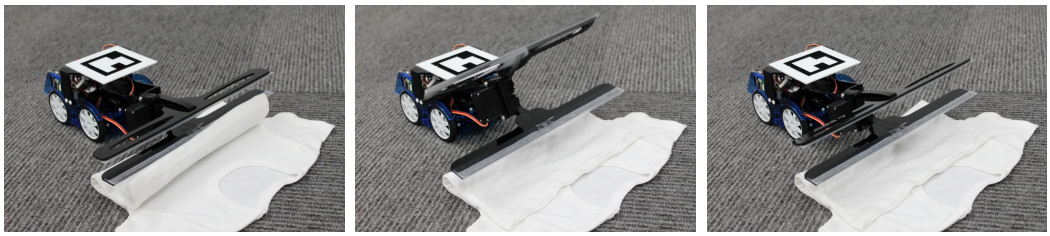
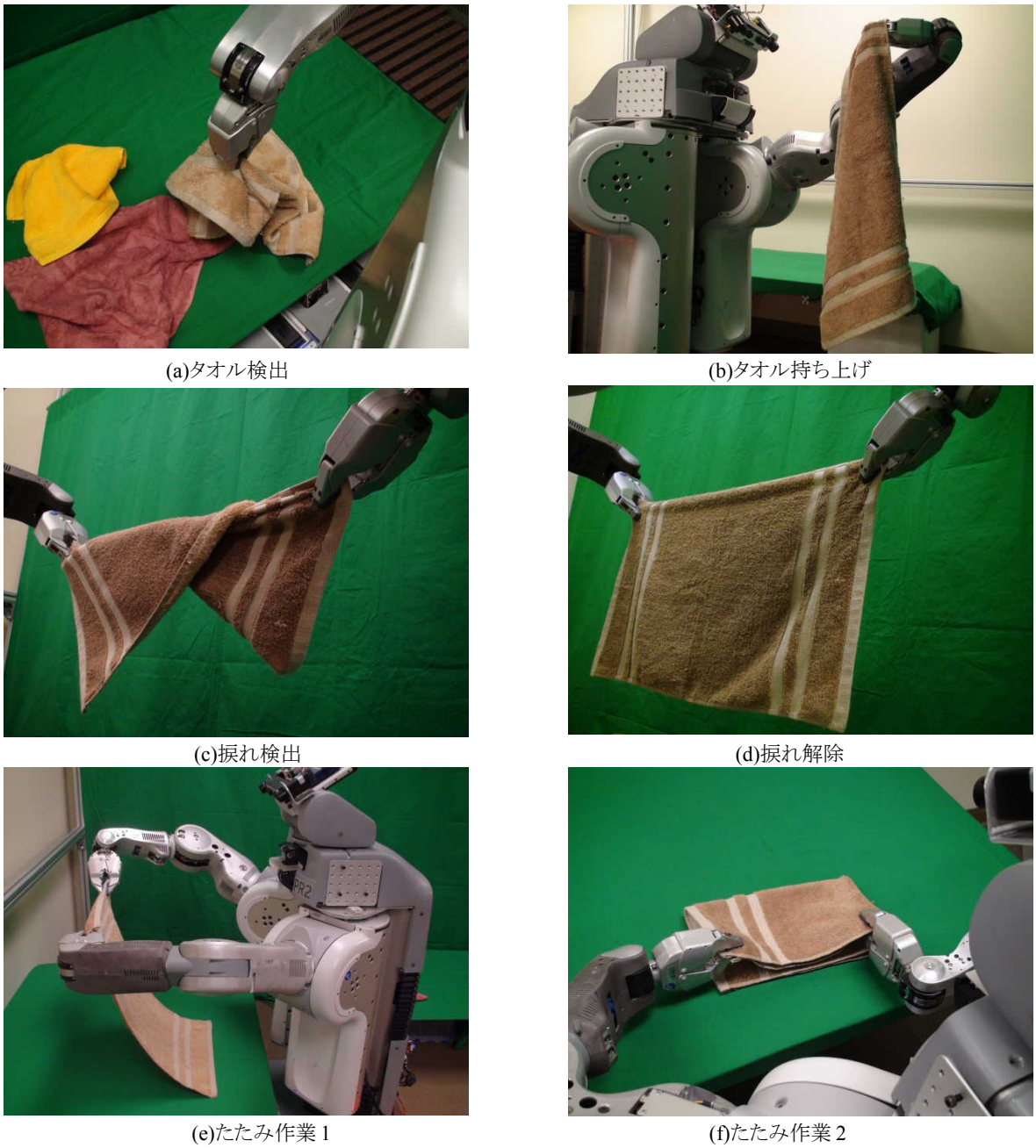


図 3.6：衣類の離脱動作

3.4 洗濯物たたみロボット

台の上に無作為に並べられたタオルを 1 枚認識し，それを 2 本のアームを用いて大きさに応じたたたみ方でたたむロボットである[9]．タオルの角を認識し，把持位置を決定する．持ち上げた際に捩れを検出し，次の把持位置を決定し掴む．それを繰り返して捩れがなくなったら，タオルの角を把持したまま台に擦るように這わせ，たたみ作業を行う．畳み終えたタオルは重ねて置く．



(a)タオル検出　(b)タオル持ち上げ

(c)捩れ検出　(d)捩れ解除

(e)たたみ作業 1　(f)たたみ作業 2

図 3.7:タオルのたたみ動作

3.5 食器洗い支援ロボット

キッチンに置かれた食器を扱いながら，食器洗い機を使った食器洗いを支援するキッチンロボットである[11]. 少子高齢社会において，単身世帯や要介護者をもつ世帯など家事の負担が大きい世帯の割合が増えることが予想されている. これに対して，食器の後片付けの負担を軽減することで少子高齢社会での家事介護支援に貢献を目的としている.

図 3.8：食器洗い支援ロボット

食器の後片付けは「食器の運搬」「残り物処理」「食器洗い」「食器収納」の 4 工程によって構築されている. これらの内，「食器洗い」を支援するロボットでキッチンロボット本体はシンクに取り付けられた，アームの先にエンドエフェクタの付いたマニピュレータである. ハンド部含めて 8 自由度の基部はスライドして移動する. 食器をつかんで持ち上げられるハンド部は開閉するだけの単純な機構だが手のひら部分に各種センサーを内蔵している. そのほか手首部分に 6 軸力センサーが取り付けられている. 食器の認識や大まかな位置の把握そのものは外界カメラで行なう. 食器等の物体をロボットが壊さないように扱うためには，多種のセンサから送られる感覚情報に瞬時に反応して動作を修正が必要である. 食器片付けのような作業目標のある動作には，注目すべきセンサ情報が限られるため，動作を支配する作業進行役が必要に応じてセンサ情報を並列的に監視させ，情報を集めながら行うべき動作を判断する方法が有効である.

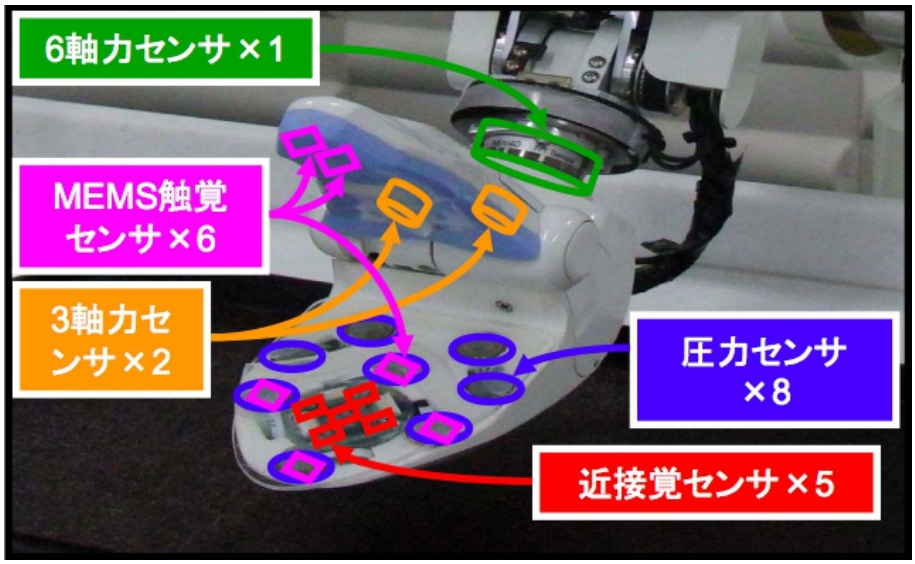


図 3.9: 多種センサが埋め込まれたエンドエフェクタ

エンドエフェクタには赤外光の反射で物体の曲率を測定できる近接覚センサーも搭載されている. これによって手をワーク対象に直接近づけることで, 把持対象物体の曲面形状を計測する. そのほか 3 軸力センサーや圧力センサーなどで取得した複数種類の感覚情報を組み合わせて, ロボットは, ならい把持動作を実施する.

2 つ目は多種センサー情報の実時間フィードバック制御技術. 多数のセンサーからの情報を利用してロボットを制御するためのソフトウェア技術だ. これまで困難だった, ロボットの動作計画を行なう上位の処理系と実時間のセンサー情報を繋ぐためのソフトウェアを開発した. 実時間でセンサー情報を処理することで自分と外界の状態を監視し, それに応じて手先の向きと動きを上位系が計画できる.

第 4 章 洗濯物自動片付けロボット

4.1 概要

本研究を行うにあたり，ロボットによる洗濯物整理の自動化に求められる処理を分析した結果，洗濯物の状態認識，洗濯物山からの取り出し処理，靴下などの小物のペアリング処理，洗濯物の片付け処理の 4 つに分割できることがわかった．また，事前のアンケートにより，シャツ，ズボンなどの大型衣類については「畳んで棚に収納する」以外に「ハンガー等でクローゼットにつるす」事で片付けと考える人も，若年層を中心に存在していた．衣類を畳む処理は，衣類の形状が不安定で取り扱いが複雑になり，ロボットによる自動化が難しい．それに対して，衣類をクローゼットにつるす処理は畳む処理と比較して，処理が単純となる事が予想されるため，大型衣類の片づけをクローゼットにつるす事とするシステムを考えた．

本研究は図 4.1 に示すような Kinect が取り付けられたクローゼット型の洗濯物片付けシステムの開発を最終目標とする．クローゼット上部の Kinect によって洗濯物や小型ロボットの状態を認識し，小型ロボットへ洗濯物の状態に応じて命令を行う．このシステムで想定する小型ロボットが行える動作は「移動」「つかむ」「離す」の 3 動作のみとする．また，ズボンや服などの大型衣類は畳まずに吊るす事で片付けとした．小物は靴下等ペアリングが必要なものはペアリングを行い，それぞれ特定の棚へ収納するものとする．本システムは関連研究で挙げたシステムとは異なり，畳まない，大型のアームを使わないことで作業の単純化，作業領域の縮小化が可能となっている．

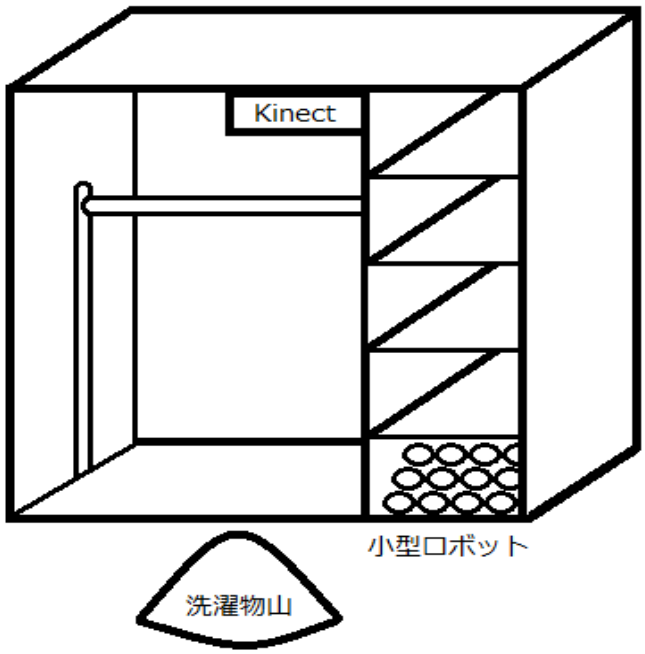


図 4.1：Kinect 付きクローゼット

4.2 想定している動作

このシステムは次のような動作で洗濯物を片付けるものと想定している.

1. 洗濯物山をシステムの認識範囲内に設置
2. 洗濯物の状態認識
3. 小型ロボットによる洗濯物山から洗濯物 1 枚の取り出し
4. 取り出された洗濯物の認識
5. 認識対象物が上着やズボンの場合, クローゼットへ収納
6. 認識対象物が靴下の場合, ペアリング処理
7. 2 で洗濯物山がないと認識されるまで 2-6 の動作の繰り返し
8. ペアリング処理済みの靴下の収納
9. 終了

洗濯物の状態認識が正確に行われていないとそれ以降の処理が確実に行われないという点から, 洗濯物の状態認識は重要であると考えられる. よって, 本研究では複数のセンサ情報を用いて洗濯物の状態を認識, その状態遷移命令を出力するシステムの開発を目的とする.

第 5 章 提案手法(1)

5.1 洗濯物の状態の定義

洗濯物の片付けにはさまざまなプロセスがあるため，ロボットが洗濯物を認識し自動で判断して片づけるためには，プロセスを複数の小さな状態に分割する必要があると考えた．．そこで本研究では，ロボットへの命令の簡単化を行うため，洗濯物片付けのプロセスを分析し，大まかに 4 つの状態を定義した．各状態の図例を図 5.1 に示す．

・第 1 状態

全種類の洗濯物が混在し，山になっている状態を第一状態とする．

・第 2 状態

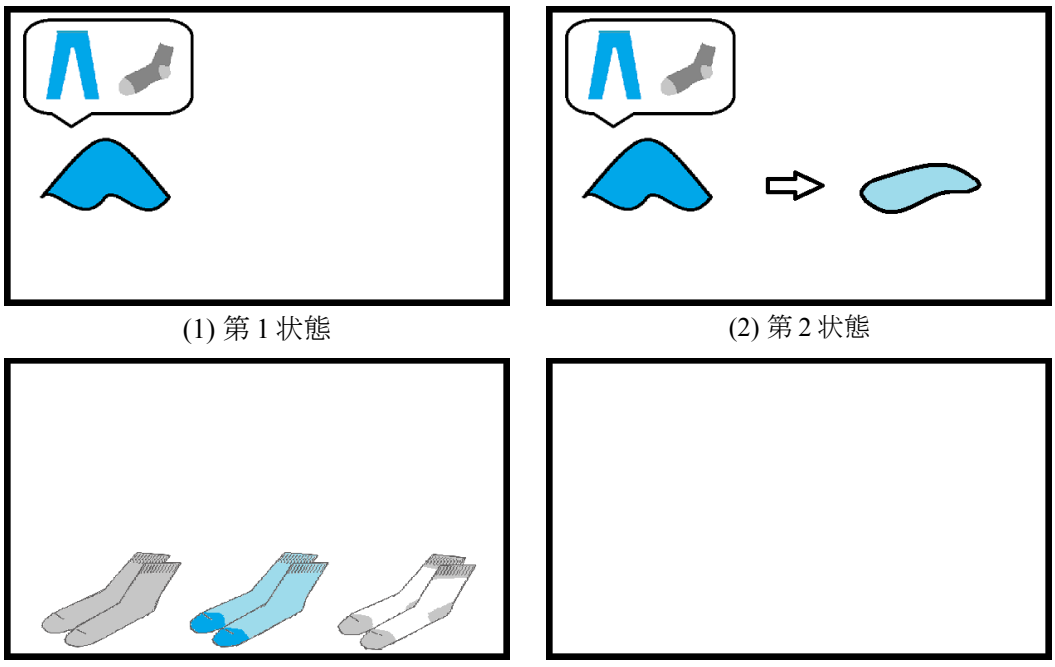
洗濯物山から洗濯物が一枚取り出されている状態を第 2 状態とする．

・第 3 状態

洗濯物のうちズボンなどの大きなものは片付けられ，ペアリングが必要な靴下等の小物の分類が完了した状態を第 3 状態とする．

・第 4 状態

全ての洗濯物が収納，整頓されている状態を第 4 状態とする．この状態は，いわゆる人間が洗濯物を片付け終えた状態と同様であるとし，洗濯物片付けは完了とする．



(1) 第 1 状態　(2) 第 2 状態

(3) 第 3 状態　(4) 第 4 状態

図 5.1：洗濯物の各状態の例

5.2 状態判定

洗濯物の状態を具体的に表現するために 4 状態に分け，それぞれの動作を IF-THEN ルールで定義した．これらを表 5.1 に示す．この定義のうち洗濯物の状態判定が正しく動作するかを実験で確かめていく．このルールによる状態遷移は図 5.2 のようになる．

表 5.1:状態遷移表

状態	対象物	IF	THEN	状態遷移	
1	洗濯物山	ある	洗濯物取り出し	状態 2	1-1
		ない	-	状態 3	1-2
2	取り出された洗濯物	ズボン	片付け処理	状態 1	2-1
		シャツ			
		靴下	ペアリング処理	状態 1	2-2
		ない	-	状態 1	2-3
3	処理済靴下	ある	片付け処理	状態 3	3-1
		ない	-	状態 4	3-2
4	洗濯物山	ある	-	状態 1	4-1
	取り出された洗濯物	ある	-	状態 2	4-2
	靴下	ある	-	状態 3	4-3
	全ての洗濯物	ない	-	終了	-

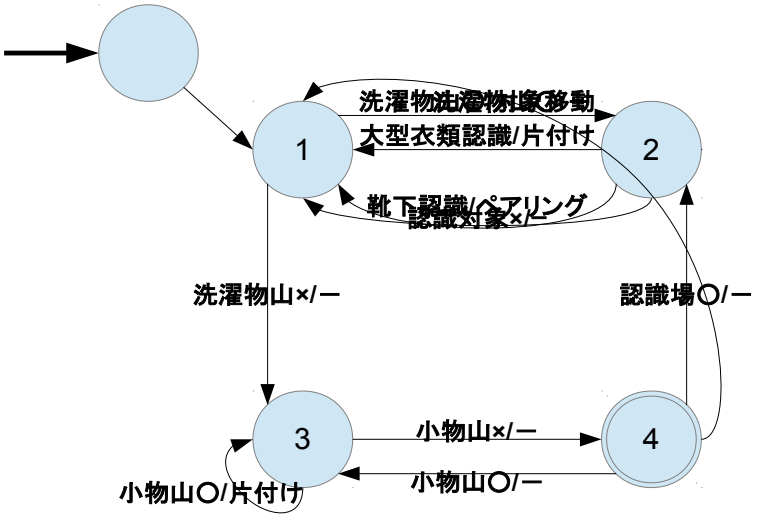


図 5.2:状態遷移図

5.3 大きさによる衣類の分類

状態 2 において，衣類の種類によって出力する命令を対応させるため，衣類の分類が必要となる．本研究では，シャツ，ズボン等の大型衣類と靴下等の小型衣類では処理が大きく異なる．そこで衣類を大まかに大型衣類と小型衣類に分けるためにカメラから見える衣類の面積による分類を提案する．

本来ならばここで衣類の種類を分類することが望ましいが，大型衣類と小型衣類で処理が大きく異なるので，シンプルで高速処理が期待できる衣類の面積による分類手法を用いて，まず，大型衣類と小型衣類の分類を試みた．以下に処理の流れを示す．

1. Kinect より得られた深度画像より背景差分法を用いて床面以外のものを抽出
2. 認識すべき対象を検出
3. 対象物の面積を求める
4. 対象物が大型衣類か小型衣類かを決定

第 6 章 実験

6.1 概要

提案手法よって状態判定と命令出力が正確に行われるか実装実験を行った．本実験は複数センサが搭載されている Kinect によるカメラ画像と深度情報を用いて洗濯物の情報を取り出して行った．今回の実験では洗濯物の分類のうち大型衣類をズボン，小型衣類を靴下の 2 種類で行うとし，ズボン 1 枚，靴下 2 枚で実験を行った．

6.2 実験方法

本実験では指定された IF THEN ルールによる洗濯物の状態の判定を重視して実験を行うため，実際のロボットは使わず洗濯物の移動は人間の手によって行った．

また，Kinect を床上 1.5m の位置にカメラセンサを下向きになるよう固定し，洗濯物を配置した．

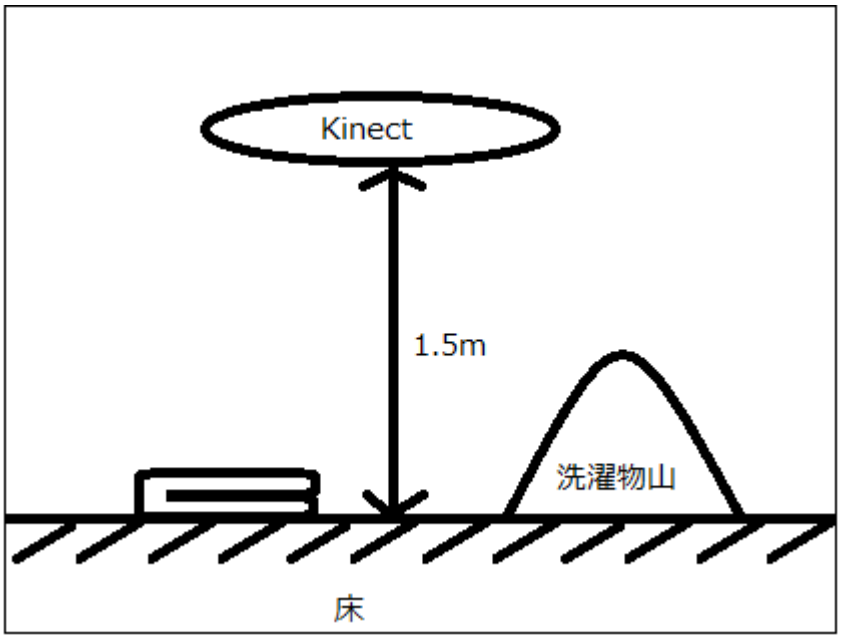


図 6.1：実験環境

6.3 実装

本実験を行うために Visual C++ 2010，RGB カメラからのカメラ画像の取得，近赤外線距離画像センサからの深度画像の取得に OpenNI，取得した画像の変換処理，表示には OpenCV を用いて行った．以下に処理手順を示す．

・実装処理

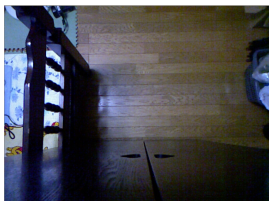
1. Kinect の初期化
2. カメラ画像，深度情報の取得
3. カメラ画像内の 3 分の 1 に小物用フィールドの生成
4. 残りの左 3 分の 1 に認識フィールドの生成
5. 残りの領域全てを洗濯物フィールドとして生成
6. 背景差分画像の生成
7. 認識フィールドの判定
8. 認識対象物が存在する場合，対象物の認識
9. 対象物の領域が 4000pixel 未満の場合小型衣類と判定
10. それ以上の場合大型衣類と判定
11. 洗濯物フィールドの判定
12. 小物フィールドの判定
13. 状態決定

6.4 実験結果

実験で行った状態の変化および状態判定結果を以下に示す．結果の評価法は状態遷移図に従って行った操作と同様の状態判定および命令出力がされているかを判定基準とした．実験で行った動作やその出力結果を表 6.1 に，その実行画面を図 6.2~6.15 に示す．

表 6.1 行った動作および判定結果

	動作	出力	
		段階認識	出力命令
1	起動	4	(開始)
2	洗濯物山配置	1	取り出し
3	靴下→認識フィールド	2	Socksペアリング
4	靴下→小物フィールド	1	取り出し
5	靴下→認識フィールド	2	Socksペアリング
6	靴下→小物フィールド	1	取り出し
7	ズボン→認識フィールド	2	Pants片付け
8	ズボン→片付け	1	取り出し
9	靴下→認識フィールド	2	Socksペアリング
10	靴下→小物フィールド	3	取り出し
11	靴下→認識フィールド	2	Socksペアリング
12	靴下→小物フィールド	3	Socks片付け



(a) RGB 画像

(b) 深度画像

(c) 各フィールドの 2 値化画像

図 6.2：表 6.1-1 行目 実行画面

図 6.2(c)は上段左が認識フィールド，右が洗濯物山フィールド，下段が小物フィールドの 2 値化画像である．

これらの図を見ると，(a)(b)ともに判定すべき洗濯物はなし，(c)の各フィールドにも対象はないため，システムは第 4 状態と正しい出力を行っている．

(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

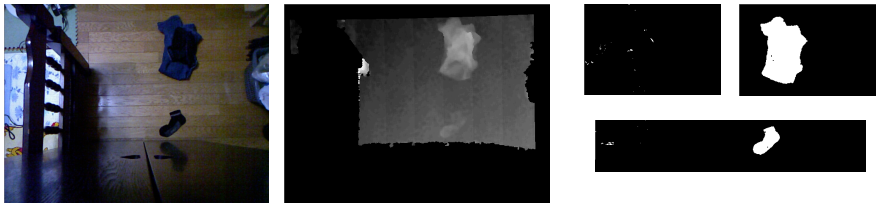
図 6.3：表 6.1-2 行目 実行画面

これらの図を見ると，洗濯物山が配置され，(c)の洗濯物山フィールドに対象がありその他フィールドには対象がないため，システムは第 1 状態と正しい出力を行っている．

(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

図 6.4：表 6.1-3 行目 実行画面

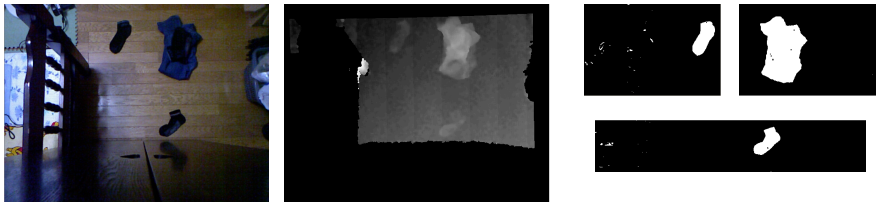
これらの図を見ると，洗濯物山から 1 枚の靴下が取り出され，(c)の認識フィールドに対象があるため，システムは第 2 状態と正しい出力を行っている．



(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

図 6.5：表 6.1-4 行目 実行画面

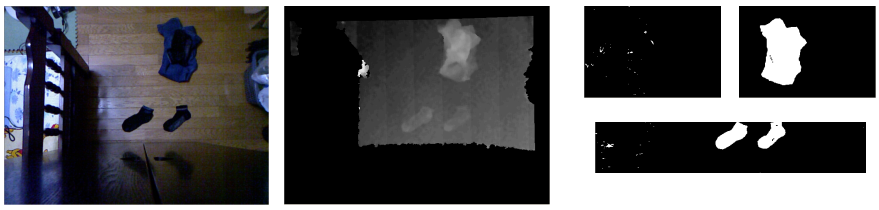
これらの図を見ると，洗濯物山からは対照が取り出されておらず，靴下が(c)小物フィールドに存在しているため，システムは第 1 状態と正しい出力を行っている．



(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

図 6.6：表 6.1-5 行目 実行画面

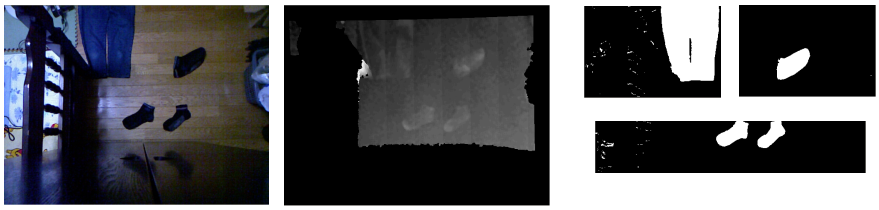
これらの図を見ると，洗濯物山が配置され，靴下が 1 枚認識フィールドに取り出されており，小物フィールドにも靴下が存在しているため，システムは第 2 状態と正しい出力を行っている．



(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

図 6.7：表 6.1-6 行目 実行画面

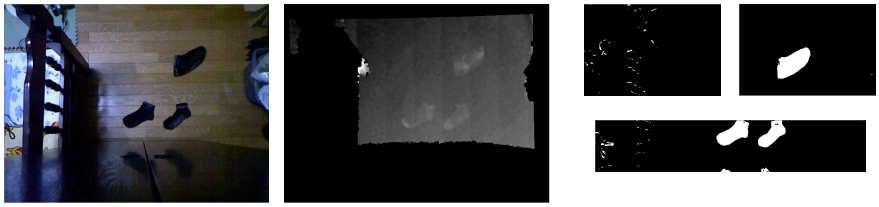
これらの図を見ると，認識フィールドの対象はなく，洗濯物山が存在し，靴下が小物フィールドに存在するため，システムは第 1 状態と正しい出力を行っている．



(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

図 6.8：表 6.1-7 行目 実行画面

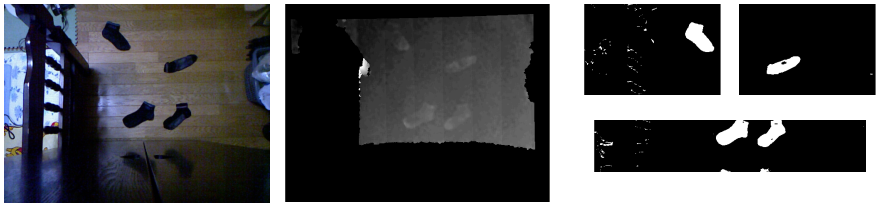
これらの図を見ると，洗濯物山からズボンが認識フィールドへ取り出され，システムは第 2 状態と正しい出力を行っている．



(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

図 6.9：表 6.1-8 行目 実行画面

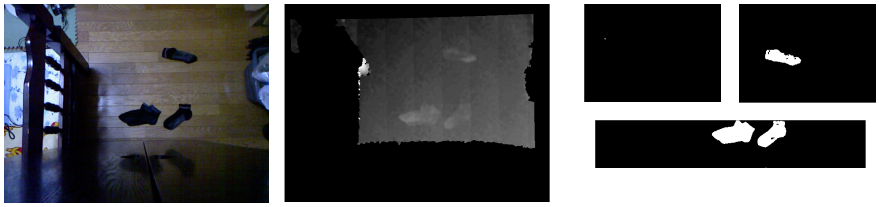
これらの図を見ると，認識フィールドに対象が存在せず，靴下が洗濯物山フィールドへ存在し，靴下が小物フィールドへ存在するため，システムは第 1 状態と正しい出力を行っている．



(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

図 6.10：表 6.1-9 行目 実行画面

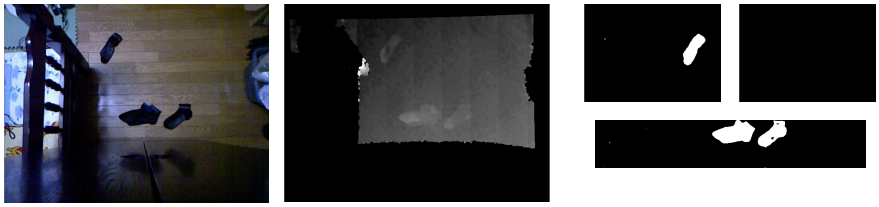
これらの図を見ると，認識フィールドに靴下が 1 枚取り出され，靴下が洗濯物山フィールドへ存在し，靴下が小物フィールドへ存在するため，システムは第 2 状態と正しい出力を行っている．



(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

図 6.11：表 6.1-10 行目 実行画面

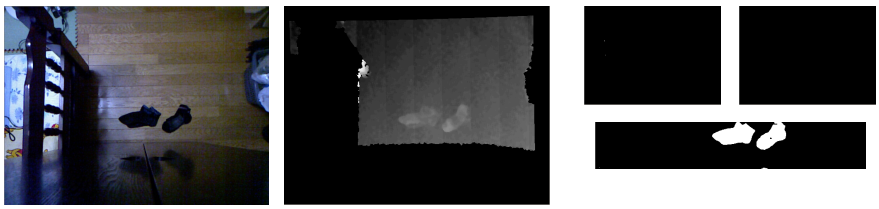
これらの図を見ると，認識フィールドに対象はなく，靴下が洗濯物山フィールドへ存在し，靴下が小物フィールドへ存在するため，システムは第 1 状態と正しい出力を行っている．



(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

図 6.12：表 6.1-11 行目 実行画面

これらの図を見ると，認識フィールドに靴下が 1 枚取り出され，洗濯物山は存在せず，靴下が小物フィールドへ存在するため，システムは第 2 状態と正しい出力を行っている．



(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

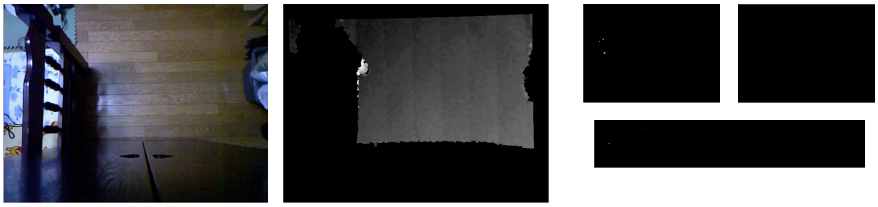
図 6.13：表 6.1-12 行目 実行画面

これらの図を見ると，認識フィールド，洗濯物山フィールドともに対象が存在せず，靴下が小物フィールドへ存在するため，システムは第 3 状態と正しい出力を行っている．

(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

図 6.14：表 6.1-13 行目 実行画面

これらの図を見ると，認識フィールド，洗濯物山フィールドともに対象が存在せず，靴下が小物フィールドへ存在するため，システムは第 3 状態と正しい出力を行っている．



(a) RGB 画像　(b) 深度画像　(c) 各フィールドの 2 値化画像

図 6.15：表 6.1-14 行目 実行画面

これらの図を見ると，(a)(b)ともに判定すべき洗濯物はなし，(c)の各フィールドにも対象はないため，システムは第 4 状態と正しい出力を行っている．

6.5 考察

提案手法の実装実験を行った結果，洗濯物の状態判定は正確に行うことができた．また，今回はズボンと靴下のみという限定的な条件であったため比較的良好な状態判定ができたと考えられる．

今後の課題として，認識するべき洗濯物の種類を実生活に用いられる程度にまで増えても，大型衣類と小型衣類の区別が可能か，処理速度は十分に高速か，等の確認がある．また，洗濯物片付けの際に通常発生しない状態を異常と判定するルールの追加や，認識できる洗濯物の種類を増やすために形状などの特徴も考慮するなど，より正確な状態遷移を行えるよう開発を進めていく．

第 7 章　洗濯物認識対象の拡張

実験 1 で述べたように，実生活では多様な種類の衣類が用いられている．実生活での運用を想定した場合，大型衣類もその種類によって処理が異なる場合が考えられる．しかし，提案手法（1）で用いた，大きさのみの判定では大型衣類の判別は非常に難しい．より多くの種類の洗濯物を認識するには画像認識や形状認識などのパターンマッチングによる画像処理技術が必要である．

7.1 テンプレートマッチング

テンプレートマッチング[11]とは，テンプレートと呼ばれる小さな一部の画像領域と同じパターンが画像全体の中に存在するかどうかを調べるパターンマッチング方法である．画像内にある対象物の位置検出，物体数のカウント，物体移動の検知などに使われる．

テンプレートマッチングの計算はテンプレートを画像の中で順番に移動させながら，テンプレートとテンプレートに重なる部分の画像の類似度を計算していく．類似度の計算は以下の 2 つの方法がある．

・SSD(Sum of Squared Difference)

$$SSD=\sum_{x=0}^{x-1}\sum_{y=0}^{y-1}(I(x,y)-T(x,y))^2$$

- 画像の輝度値を I(x,y)
- テンプレート画像の輝度値を T(x,y)

・SAD(Sum of Absolute Difference)

$$SAD=\sum_{x=0}^{x-1}\sum_{y=0}^{y-1}(I(x,y)-T(x,y))$$

- 値が小さいほど，テンプレートとテンプレートに連なる部分の画像が似ている
- 完全に一致している場合には，値は 0 になる

7.2 物体認識

物体認識[12]は，大きく分けて二種類の技術が存在し，一つは identification，もう一つが classification となる．Identification とは物体同定のことであり，認識対象の物体を一つに定め，認識対象が撮影された画像を positive data，認識対象が撮影されていない画像を negative data として学習させ，画像が入力された際に，対象画像に含まれる物体が認識対象の物体であるかどうかの真偽判定を行う．

対して classification とは，物体分類のことを指し，事前に認識対象となる物体の数とラベル（物体の名称）を定めておき，学習状態では，カテゴリー別に学習を行う．そして認識状態では，画像を入力として，その画像に撮影された物体が該当するラベル番号（物体名称）を出力する．

一般的に，物体認識というと identification のことを指すが，一般物体認識というと classification のことを指す．

7.2.1 Bag of Keypoints 法

Bag of keypoints 法[13]は，統計的言語処理における Bag of words のアナロジーで，テキスト表現手法の Bag of words の考え方を画像に適用させた手法となる．bag of words が，語順を無視して文章を単語の集合と考えるのと同様に，bag of keypoints では位置を無視して，画像を局所特徴(keypoints)の集合として考える．

全体の流れは，

1. 画像の局所特徴を表現する局所特徴ベクトルである keypoints を画像から抽出する．この局所特徴ベクトルの表現には SIFT 法が用いられることが一般的となっている．

2. 得られた keypoints に対して，ベクトル量子化を行う．ベクトル量子化に最もよく使われている手法は k-means 法であり，クラスタリングされた各クラスの銃身を，代表ベクトルとして生成する．この各クラスタの銃身である代表ベクトルを visual word と呼ぶ．

3. visual word をまとめたものを code book と呼び，codebook を用いて画像の特徴ベクトルを生成する．

4. 学習画像から生成された画像の特徴ベクトルを学習データ，テスト画像から生成された画像の特徴ベクトルをテストデータとして，分類器による学習と識別をおこなう．よく使われる手法として SVM がある．

という流れになる．

つまり，bag of keypoints では，画像の特徴良は，画像の局所特徴から生成した代表ベクトルの出現頻度のヒストグラムによって表現される．

7.2.2 SIFT(Scale Invariant Feature Transformation)

SIFT[14]は，ある特徴点の代表輝度勾配方向を決定し，その方向を基準として，他方向の輝度勾配ヒストグラムを作成し，多次元ベクトルで特徴量を記述する．これにより，回転・スケール変化・照明変化に頑強であるという特徴を持つ．

SIFT 特徴点抽出の流れは，特徴の抽出に適した点の検出と，特徴量の記述の 2 状態からなり，検出状態はスケールスペース極値検出，キーポイントのローカライズからなり，記述状態はオリエンテーション割り当てと，SIFT Descriptor による特徴量の抽出からなる．スケールスペース極値検出によるスケール変化に，オリエンテーションの割り当てによる回転に，SIFT Descriptor による特徴量の記述による証明変化に，それぞれ不変であるという特徴を得る．

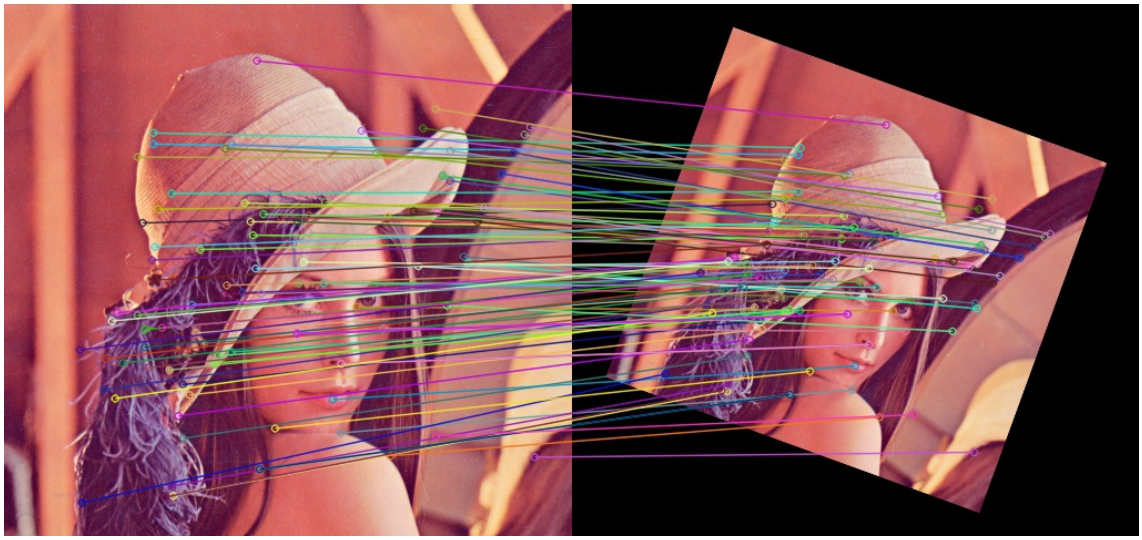


図 7.1：SIFT 特徴量による対応点検出

7.2.3 k-means

Bag of keypoints 法におけるベクトル量子化には，k-means 法[15]によるクラスタリングによって生成した codebook と呼ばれる visualword の集合を用いる．

7.2.3.1 クラスタリング

クラスタリングとは，さまざまなパターンを持った入力データを，類似したパターンごとに別々のクラスに分類する技術となる．クラスタリング手法は，階層的手法と，k-means などの非階層的手法に分けられる．

7.2.3.2 階層的クラスタリング法

階層的手法は，1 個の要素だけを含む N 個のクラスタがある初期状態から，クラスタ間の距離関数に基づいて，最も距離の近いクラスタをマージする．そして，この処理を全ての要素が一つのクラスタに統合されるまで繰り返すことで階層構造を生成する．

この階層構造はデンドログラムによって表現される．デンドログラムでは，各終端ノードが各要素を表し，マージされてできたクラスタを非終端ノードで表した 2 分木となっている．

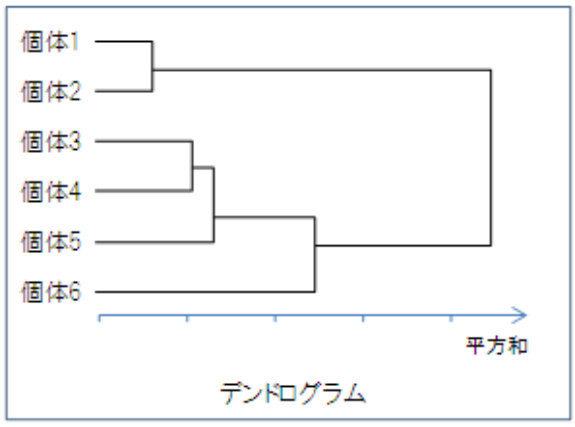


図 7.2：デンドログラムの例

また，クラスタ C1 と C2 の距離関数 D(C1,C2) の違いにより以下のような手法に分類される．

・最短距離法 or 単連結法

$$D(C_1, C_2) = \min_{x \ni C_1, x2 \ni C_2} D(x_1, x_2)$$

・最長距離法 or 完全連結法

$$D(C_{1,}C_2)=\max_{x\ni C_{1,}x2\ni C_2} D(x_{1,}x_2)$$

・群平均法

$$D(C_{1,}C_2)=\frac{1}{n_1 n_2}\sum_{x_1\ni C_1}\sum_{x_2\ni C_2} D(x1,x2)$$

・ウォード法

$$D(C_{1,}C_2)=E(C_1\cup C_2)-E(C_1)-E(C_2)$$

ただし $$E(C_i)=\sum_{x\ni C_i}(D(x,c_i))^2$$

ウォード法は，各要素からその要素を含むクラスタの重心までの距離の二乗の総和を最小化する．また，最短距離法，最長距離法，群平均法は任意の要素間の距離 D が与えられている場合に適用可能となる．要素がベクトルで表現されている場合は，ベクトル間のユークリッド距離などを求め，適用する．

クラスター分析におけるベクトル間の距離計算には以下のものが用いられる．

・ユークリッド距離

・重み付きユークリッド距離

・マハラノビス距離

・ミンコフスキー距離

7.2.3.3 非階層的クラスタリング法

非階層的クラスタリング法では，分割の良し悪しの評価関数を定め，その評価関数を最適にする分割を探索する．可能な分割の総数は N に対して指数的となるため，実際は準最適解を求めることになる．

代表的な非階層的手法である k-means 法は，重心をクラスタの代表点とし，評価関数を最小化する．次式がその評価関数となる．

$$\sum_{i=1}^{k}\sum_{x\ni C_1}(D(x,c_1))^2$$

最適解の探索は各要素のクラスタへの割り当てと代表点の再計算を繰り返し行う．この手法は山登り法で，局所最適解しか求められないため，ランダムに初期値を変更して，評価関数が最小となる結果を選択する．

k-means 法の処理手順は次の 3 ステップで表すことができる.

1. k 個の代表点をランダムに選択.

2.全ての要素を, 評価関数が最小となる代表点に割り当てる.

3.代表点への割り当てが変化しない場合終了. 変化した場合, 各クラスタの重心を新たな代表点として 2 へ戻る.

k-means 法の利点としては, データの入力順序に結果が依存しないこと, 次元数・データ数に対するスケーラビリティがあること, 特定のクラスタ数における各クラスタサイズを最小化できること, クラスタの境界が明確に決定すること, などが挙げられる.

k-means 法の欠点としては, 多用なクラスタ形状を扱えないこと, 実数値データ, 整数値データ, カテゴリデータなど, 異なる属性を同時に扱えないこと, 最適なクラスタ数の発見やクラスタの境界が非線形な問題の扱いが困難であること, 初期値に結果が依存してしまうため, 異なる初期設定により異なる結果となりうることなどが挙げられる.

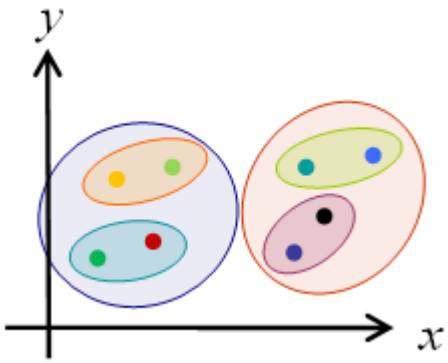
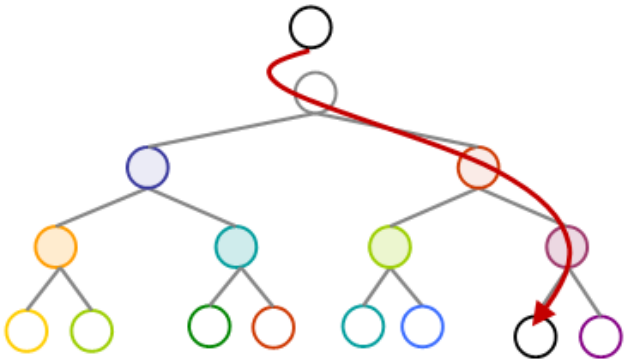


図 7.3 : k-means によるクラスタリング



K-Means tree

図 7.4 : 階層的 k-means tree

7.2.4 SVM

SVM[16]はニューロンのモデルとして最も単純な線形しきい素子を用いて，2 クラスの識別器を構成する手法である．1960 年代に vapnik らが考案した Optimal Separating Hyperplane を起源とし，カーネル学習法と組み合わせると非線形の識別器になる．この拡張はカーネルトリックと呼ばれる手法で，このカーネルトリックの功績で，現在知られている多くの手法の中でも認識性能の優れた学習モデルの一つであると考えられている．

SVM は基本的に 2 クラス識別にしか対応していないため，複数クラスで識別する場合には，複数の SVM を組み合わせるなどの工夫が必要となる．SVM を内部的に組み合わせて，複数クラスの分類を行ってくれるプログラムとして，LIBSVM や，SVMlight の複数クラス分類版である SVMmulticlass などがある．

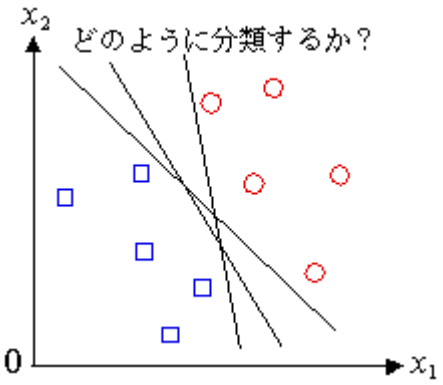


図 7.5:2 次元データの分類

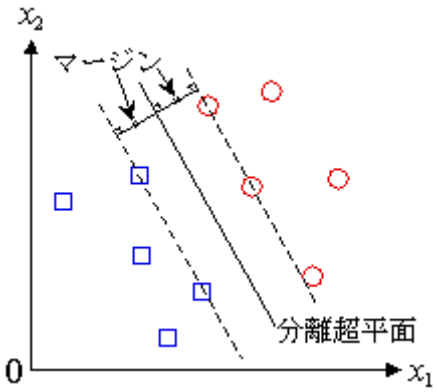


図 7.6:マージン最大化

第 8 章提案手法(2)

8.1 概要

5 章では，衣類の分類のために大きさ(カメラから見える面積)を用いた．これは，洗濯物に折れがなく，完全に広げられている状態という理想的な環境を想定し，大型衣類と小型衣類の分類を行った．しかし，大型衣類には上着やタオル，下着など多くの種類の衣類が存在しており，認識の際に折れが発生してる．同じ大型衣類でもケースに応じて異なる処理を行う場合が考えられる．そこで我々は，衣類の分類に用いる情報を大きさだけでなく，形状情報を用いることで，認識できる衣類の種類を増やし，認識の柔軟さの向上を図る手法を提案する．

8.2 体積による分類

前述のように衣類の分類に大きさとしてカメラから見える面積を用いていたが，もし衣類に折れが発生していた場合，見える面積は折れている分だけ小さくなる．この問題に対処するために，衣類の体積を用いた分類を提案する．

Kinect より得られる深度センサから得られる高さ情報からカメラ画像内の衣類と思われる部分のみを合計することで衣類の体積を大まかに計算することができる．

8.3 形状情報による分類

体積のみによる分類では，5 章で行った大型衣類，小型衣類の分類に対しては有効であるが，さらに細かく分類する場合に対応できない．そこで，大きさによる分類だけでなく形状情報を用いることで 5 章よりも多種の衣類に対応することができると考えた．

ここでいう形状情報は Kinect の深度情報より得られた分類対象の 2 値化画像のことである．(図 8.1,8.2)

このような情報を教師データとして，bag of keypoints 法を用いて分類ごとにいくつか学習させておく．これらの情報と，入力された画像とのテンプレートマッチングを行い，最も似ている画像の所属クラスがその衣類の分類となる．

(a)深度画像　　(b)2 値化画像

図 8.1:シャツの形状情報例 1

(a)深度画像　　(b)2 値化画像

図 8.2:シャツの形状情報例 2

8.4 アルゴリズム

上記 2 つの手法を用いて実際にどのように分類を行うかを以下に示す.

1. Kinect より認識対象物の抽出
2. 認識対象物の体積の計算
3. 体積によって大まかに大きいものか小さいものか決定
4. bag of keypoints 法を用いたテンプレートマッチング
5. 所属クラスの決定

第 9 章　おわりに

本研究では，Kinect によるカメラ画像と深度情報を用いた洗濯物状態判定システムを提案した．

提案手法 1 の実装実験を行った結果，洗濯物の状態判定は正確に行うことができた．また，今回はズボンと靴下のみという限定的な条件であったため比較的良好な状態判定ができたと考えられる．提案手法 2 として衣類の変形を考慮し，面積ではなく大まかな体積を用いることで衣類の大小の大まかな分類がより正確に行われると考えた．また，衣類の種類の増加に対応するため，形状の情報を用いたテンプレートマッチングの手法を考えた．これらの 2 つの情報を用いることでより柔軟な衣類の分類が期待できる．

今後の課題として，認識するべき洗濯物の種類を実生活に用いられる程度にまで増えても，大型衣類と小型衣類の区別が可能か，処理速度は十分に高速か，等の確認がある．また，洗濯物片付けの際に通常発生しない状態を異常と判定するルールの追加や，認識できる洗濯物の種類を増やすために形状などの特徴も考慮するなど，より正確な状態遷移を行えるよう開発を進めていく．また，提案手法 2 の有効性の検証と検討をする必要がある．

謝辞

本論文作成に当たって日々ご指導頂いた三好力教授には深く感謝いたします．また，様々なアドバイスや相談を下さった同三好研究室の皆様や，友人の皆様に深く感謝いたします．

参考文献

[1]「どんなサービスがあるの？- 訪問介護（ホームヘルプ）| 公表されている介護サービスについて| 介護事業所検索「介護サービス情報公表システム」」

http://www.kaigokensaku.jp/publish/group2.html

[2]「訪問サービス―訪問介護―サービス内容―生活援助（家事援助）- ［介護］介護保険」

http://kaigo.k-solution.info/2008/05/_1_179.html

[3] "iRobot Corporation"

http://www.irobot.com/

[4] "ECOVAS ROBOTICS"

http://www.ecovacs-japan.com/

[5] “Kinect - Xbox.com”

http://www.xbox.com/ja-JP/kinect

[6] 根津禎，“Kinectに見るジェスチャー入力の可能性”，日経エレクトロニクス（2010.12.27）

[7]「ホームアシスタントロボットによる掃除後片付けを行う技術」

http://www.irt.i.u-tokyo.ac.jp/reform/081024/index.shtml

[8] 杉浦裕太，”家庭用ロボットのための直観的な支持手法に関する研究”，慶應義塾大学大学院メディアデザイン研究科修士論文(2009)

[9] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei and Pieter Abbeel, "Cloth Grasp Point Detection based on Multiple-View Geometric Cues with Application to Robotic Towel Folding", In IEEE International Conference on Robotics and Automation(2010).

[10] キッチンロボットにより食器洗いを支援する技術

http://www.irt.i.u-tokyo.ac.jp/reform/081217/index.shtml

[11] 基本的なテンプレートマッチング

http://isl.sist.chukyo-u.ac.jp/Archives/tm.html

[12] 柳井啓司,"一般物体認識の現状と今後", 情報処理学会論文誌, vol.48, No. SIG16(CVIM 19).

[13] G. Csurka, C. Dance, L. Fan, and C. Bray, “Visual categorization with bags of keypoints,” ECCV, pp. 1-22, 2004.

[14] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” International Journal of Computer Vision, Vol.60, No.2, pp.91-110, Jan.2004

[15] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability", Berkeley, University of California Press, 1:281-297, 1967.

[16] Corinna Cortes and V. Vapnik, "Support-Vector Networks", Machine Learning, 20, 1995.

[17] 中村薫，“Kinect センサープログラミング”，秀和システム（2011.06.01）

[18] “OpenCV - 逆引きリファレンス”
http://opencv.jp/cookbook/index.html
[19] “OpenNI とMS SDK の優劣”
http://itpro.nikkeibp.co.jp/article/COLUMN/20111114/374448/

学会発表履歴

[1] 広瀬大樹, 三好力, 3D 情報を用いた洗濯物片付き度合い判定システムの提案, 情報処理学会第 76 回全国大会, 2014 年 3 月 11 日‐13 日.

[2] 広瀬大樹, 三好力, 3D 情報を用いた洗濯物の状態判定, FIT2014 第 13 回情報科学技術フォーラム, 2014 年 9 月 3 日‐5 日.

[3] Daiki Hirose , Tsutomu MIYOSHI, Kazuki Maiya, "A Study of Laundry Tidiness: Laundry State Determination Using Video and 3D Sensors", ICSIIT2015, (2015-03 to appear).

付録

```
#include <opencv2/opencv.hpp>

#ifdef _DEBUG

    //Debug モードの場合

    #pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_core220d.lib")          //
opencv_core

    #pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_imgproc220d.lib")       //
opencv_imgproc

    #pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_highgui220d.lib")       //
opencv_highgui

    #pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_objdetect220d.lib")    //
opencv_objdetect

    //以下, 必要に応じて追加

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_ml220d.lib")          // opencv_ml

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_features2d220d.lib")    //
opencv_features2d

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_video220d.lib")       //
opencv_video

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_calib3d220d.lib")       //
opencv_calib3d

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_flann220d.lib")       //
opencv_flann

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_contrib220d.lib")       //
opencv_contrib

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_legacy220d.lib")       //
opencv_legacy

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_gpu220d.lib")          //
opencv_gpu

#else

    //Release モードの場合

    #pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_core220.lib")          //
opencv_core

    #pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_imgproc220.lib")       //
opencv_imgproc

    #pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_highgui220.lib")       //
opencv_highgui

    #pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_objdetect220.lib")    //
opencv_objdetect

    //以下, 必要に応じて追加

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_ml220.lib")          // opencv_ml

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_features2d220.lib")    //
opencv_features2d

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_video220.lib")       //
opencv_video

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_calib3d220.lib")       //
opencv_calib3d

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_flann220.lib")       //
opencv_flann

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_contrib220.lib")       //
opencv_contrib

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_legacy220.lib")       //
opencv_legacy

    //#pragma comment(lib,"C:\\OpenCV2.2\\lib\\opencv_gpu220.lib")          //
opencv_gpu

#endif
```

```
#include <XnCppWrapper.h>

#pragma comment(lib,"C:/Program files/OpenNI/Lib/openNI.lib")


#define SAMPLE_XML_PATH "C:/Program Files/OpenNI/Data/SamplesConfig.xml"

using namespace cv;

using namespace xn;


int determinationLaundry(int laundry,int determin,int smaller);


int state = 0;


int main()

{

    Context context;

    EnumerationErrors errors;


    context.InitFromXmlFile(SAMPLE_XML_PATH);


    DepthGenerator depthGenerator;// depth context

    context.FindExistingNode(XN_NODE_TYPE_DEPTH, depthGenerator);


    ImageGenerator imageGenerator;//image context

    context.FindExistingNode(XN_NODE_TYPE_IMAGE, imageGenerator);



    DepthMetaData depthMD;

    ImageMetaData imageMD;


    Mat image(480,640,CV_8UC3);

    Mat depth(480,640,CV_16UC1);


        Mat depth_rw(480,640,CV_8UC1);


        Mat background1,diff1,gry1,output1;//差分用

        Mat background2,diff2,gry2,output2;//差分用


        Mat gray1,gray2;


        Mat depth_img,dst_img1,dst_img2;//加工用

        Mat dpdst_img1,dpdst_img2;//加工用


        Mat roi_laundry,roi_determin,roi_smaller;//切り抜き用


        Mat initmask(480,640,CV_8UC1);


        int key = 0;
```

```
        int i,j,k;
//      double dis_rw;
        const char th = 30; //閾値
    bool isWarp=false;

        unsigned short* dp = (unsigned short*)depth.data;
        unsigned short rdp;

    while (key!='q')
    {
        //wait and error processing
        context.WaitAnyUpdateAll();

        imageGenerator.GetMetaData(imageMD);//カメラ生データ取得
        depthGenerator.GetMetaData(depthMD);//深度生データ取得
        depthGenerator.GetAlternativeViewPointCap().SetViewPoint(imageGenerator);//ズレを補正

        memcpy(image.data,imageMD.Data(),image.step * image.rows);    //イメージデータを格納
        memcpy(depth.data,depthMD.Data(),depth.step * depth.rows);    //深度データを格納

                        for(k=0,i=0;i<depth_rw.rows;i++){
                                for(j=0;j<depth_rw.cols;j++,k++){
                                        rdp = depthMD[k];
                                        if( rdp > 1600){
depth_rw.data[i*depth_rw.cols + j] = 0;
                                        }
                                        else if( rdp > 1300){
                                                rdp = (unsigned short)((double)((double)((1600 - rdp)/300.0) * 255));
depth_rw.data[i*depth_rw.cols+j] = (unsigned char)rdp;
                                        }
                                        else{
depth_rw.data[i*depth_rw.cols + j] = 0;
                                        }
                                }
                        }

                        if (background1.empty()){
                                background1 = image.clone();
                                //background = depth_img.clone();
                        }
                        if (background2.empty()){
                                background2 = depth_rw.clone();
                                //background = depth_img.clone();
                        }

                        absdiff(image,background1,diff1);//背景画像とカメラ画像の差分を取得
        cvtColor(diff1,gry1,CV_BGR2GRAY);//差分画像をグレイスケールに

        threshold(gry1, gry1, th, 255, THRESH_BINARY);//二値化画像化

                        resize(gry1, dst_img1, cv::Size(), 0.5, 0.5,INTER_AREA);
                        resize(gry1, dst_img1, cv::Size(), 0.5, 0.5,INTER_AREA);
                        resize(dst_img1, dst_img2, cv::Size(), 2, 2);
                        resize(dst_img1, dst_img2, cv::Size(), 2, 2);

                        output1 = dst_img2.clone();
                        threshold(output1, output1, 200, 255, THRESH_BINARY);//二値化画像化

                        absdiff(depth_rw,background2,diff2);//背景画像と深度画像の差分を取得

                        threshold(diff2, diff2, 10, 255, THRESH_BINARY);//二値化画像化

                        resize(diff2, dpdst_img1, cv::Size(), 0.5, 0.5,INTER_AREA);
                        resize(diff2, dpdst_img1, cv::Size(), 0.5, 0.5,INTER_AREA);
                        resize(diff2, dpdst_img1, cv::Size(), 0.5, 0.5,INTER_AREA);
                        resize(dpdst_img1, dpdst_img2, cv::Size(), 2, 2);
                        resize(dpdst_img1, dpdst_img2, cv::Size(), 2, 2);
                        resize(diff2, dpdst_img1, cv::Size(), 0.5, 0.5,INTER_AREA);
                        resize(dpdst_img1, dpdst_img2, cv::Size(), 2, 2);
                        resize(dpdst_img1, dpdst_img2, cv::Size(), 2, 2);

                        output2 = dpdst_img2.clone();
                        threshold(output2, output2, 200, 255, THRESH_BINARY);//二値化画像化

                        roi_laundry = output1(Rect(320,0,320,240));
                        roi_determin = output1(Rect(0,0,320,240));
                        roi_smaller = output1(Rect(0,240,640,240));

                        //convert color space RGB2BGR
        cvtColor(image,image,CV_RGB2BGR);

        imshow("image",image);
        imshow("depth",depth);
                        imshow("output1",output1);
                        imshow("output2",output2);

                        imshow("depth_rw",depth_rw);

                        imshow("laundry",roi_laundry);
                        imshow("determination",roi_determin);
                        imshow("smaller",roi_smaller);
```

```
                if (key == 's'){
                        //background1 = image.clone();
                        background2 = depth_rw.clone();

                        printf("center:
%d[mm]\n",depthMD[depth.cols*depth.rows/2 + depth.cols/2]);
                        //background = depth_img.clone();
                }
                else if(key == 'd'){
                        if(state == 0){
                                state++;
                        }
                        else{
                        printf("State is
%d\n",determinationLaundry(countNonZero(roi_laundry),countNonZero(roi_determin),co
untNonZero(roi_smaller)));
                        }
                        /*printf("laundry_
%d\n",countNonZero(roi_laundry));
                        printf("determin_
%d\n",countNonZero(roi_determin));
                        printf("smaller_
%d\n",countNonZero(roi_smaller));*/
                }

        memcpy(depth_rw.data,initmask.data,depth_rw.step*depth_rw.rows);

                key = waitKey(33);
    }
    context.Shutdown();
    return 0;
}

int determinationLaundry(int laundry,int determin,int smaller){
        int ret = 4;

        if( determin > 500){
                ret = 2;
                printf("determin = %d\n",determin);
                if(determin >4000){
                        printf("pants\n");
                }
                else{
                        printf("socks\n");
                }
        }
        else if(laundry > 500){
                ret = 1;
        }
        else if( smaller > 500){
                ret = 3;
        }

        return ret;
}
```