平成 26 年度　修士論文

複数センサを利用した
衣類識別に関する研究

龍谷大学　大学院

理工学研究科　情報メディア学専攻

T13M074　米谷 和記

指導教員　三好 力　教授

内容梗概

近年，ロボット技術の進歩にともない，家事代行ロボットの普及が進んでいる．ここでの家事の定義は，介護保険サービスより掃除，料理，買い物，洗濯，衣類整理としているが，それぞれに代行サービスが存在している．iRobot 社の Roomba に代表される掃除ロボットも一般的なものとなってきている．しかし，衣類整理に当たる「洗濯物を片付けるロボット」はあまり一般的ではない．これは，対象物である洗濯物が衣類であり，操作するたびに多種多様な形状に変化し，定型的な操作の繰り返しによる取り扱いが難しいためであると考えられる．

本研究を行うにあたり，ロボットによる洗濯物の片づけに求められる機能を分析した結果，洗濯物の分類，靴下などの小物のペアリング，収納処理に分割できることがわかった．この内靴下はペアリングされていないと使用できず，作業に手間がかかる点から，靴下のペアリングは重要度が高いと考える．よって我々は，距離情報と画像処理を用いることで靴下の折れを検知・修正し，対象画像を分割することでより精度の高いペアリングを行うシステムを開発した．

Abstract

Recent years, house-robots have become popular with the advances in robot technology, but there is no alternative for the tidiness of laundry by robots. We analyzed the functions required to tidy up laundry by robots, and thought that the pairing of socks is important. The purpose of this study is to develop a system for pairing socks automatically by robots. This paper provides the recognition part of this system that includes deforming detection and the similarity determination of socks by the use of image processing and 3D information. The results show that the system could detect deforming position and gripping point correctly, and could classify socks of same pattern into same group. The study concludes that the method to revise the deformation using multiple sensors is effective.

目次

第 1 章　はじめに

1.1　研究背景

近年，インターネットの普及やロボット技術の進歩により，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[1]．これにより，複数の家事を並列的に行うことが可能となり，時間的・身体的負担の軽減に貢献している．

また，一言で「家事」といっても様々なものが存在するが，本論文での「家事」とは，介護サービスの定義より，掃除，料理，買い物，洗濯，衣類整理としている[2][3]．介護サービスはこの 5 つをまとめて代行するサービスであるが，家事によっては個別の自動化・代行サービスが存在する．図 1 に家庭用の代行サービスの一例を示す．掃除であれば iRobot 社の Roomba[4]に代表される掃除ロボットも一般的なものとなってきている．最近では床の掃除だけでなく，窓の掃除も可能なタイプの掃除ロボット[5]も登場しており，自動化できる範囲は広がり続けている．料理の代行は，冷凍食品を電子調理器で温める，出前をとるといったことで可能である．食器の洗浄も食器洗浄機が存在している．通信販売サイトと宅配サービスの利用によって，家から出ることなく商品を入手することができるし，洗濯に関しても洗濯乾燥機を用いることで自動化されている．しかし，洗濯した後の「衣類整理」にあたる自動化・代行サービスは，クリーニングサービスといった家庭用ではないものが多く，一般家庭に取り入れるには巨大・高価である．これは，対象物である洗濯物が衣類であるため，操作するたびに多種多様な形状に変化し，機械による取り扱いが難しいためであると考えられる．

表 1：家庭用の代行サービス例

掃除	料理	買い物	洗濯	衣類整理
				

1.2 研究目的

現在，洗濯物の整理をロボットなどの機械に行わせることは前述のようにあまり一般的ではない．洗濯物整理の自動化が実現すれば，高齢者などの足腰の弱い人の身体的負担の軽減や，共働きの家庭など多忙な人の時間的負担の軽減が望める．そこで，洗濯物の整理をロボットに行わせることを最終目的とし研究を行う．

本研究を行うにあたり，ロボットによる洗濯物の整理に求められる処理を分析した結果，洗濯物の分離処理，靴下などの小物のペアリング処理，収納処理に分割できることがわかった．この中で，靴下はペアリングされていないと使用できない，作業に手間がかかるといった点から，靴下のペアリングは重要度が高いと考える．よって，3D 情報と画像処理を用いることで靴下の折れを検知，修正しペアリングを行うシステムを開発する．今回，情報を取得するために用いたデバイスは，RGB カメラセンサと距離画像センサを搭載しており，比較的安価な Kinect[6]である．

第 2 章　Kinect とは

Kinect[6]は Microsoft 社から販売されている家庭用据え置き型ゲーム機 Xbox360 向け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．特徴として Kinect が内蔵するセンサによってプレイヤーの動きを捉え，コントローラを使用せずにゲームの操作を可能とすることが挙げられる．このような音声やジェスチャーなど「人の自然な動作」で入力を行うユーザインタフェースのことを Natural User Interface(NUI)という．Kinect には，3 つのセンサおよび専用ソフトウェアを動作させるプロセッサが内蔵されており，人間を検出しプレイヤーの姿勢や動きを認識することができる．これにより従来のようなボタン操作ではなく，プレイヤー自身の体を使って直観的にゲームを操作することが可能となった．内蔵されているセンサは，写真撮影や映像録画に用いられる RGB カラー映像認識用カメラセンサ，Kinect から対象物までの距離を測定するための近赤外線距離画像センサ，音声を認識するためのマイクセンサの 3 種類である．さらにユーザの動きに対応するため，上下の角度を自動調整する電動チルト機構も備えている．

Kinect は人物の動きをデジタル的に記録するモーションキャプチャ技術を使用しているが，上記のセンサを用いることで，キャプチャ時に着る特殊なスーツおよび検出時に使用するトラッカーが必要なくなっている．これによりカメラに被写体を映す事でプレイヤーから Kinect 本体までの距離を計測し，ユーザの骨格のさまざまな動きを検出，ゲーム内のキャラクターの動きにリアルタイムに反映させることが可能となっている．

図 1：Kinect 外観

対象物との距離を計測する近赤外線距離画像センサは，赤外線を照射する近赤外レーザとレーザ照射パターンを捉えるための近赤外カメラで構成されている．また，計測方法として三角測量を利用している．具体的にまず，Kinect に搭載した近赤外レーザから対象物にスポット光を複数，ランダムに配列したようなパターンを投影し，対象物に映ったパターンをカメラで捉える．そして，捉えたパターンのカメラからの見込み角を求める．これに，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して，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点の距離計測が可能になる．

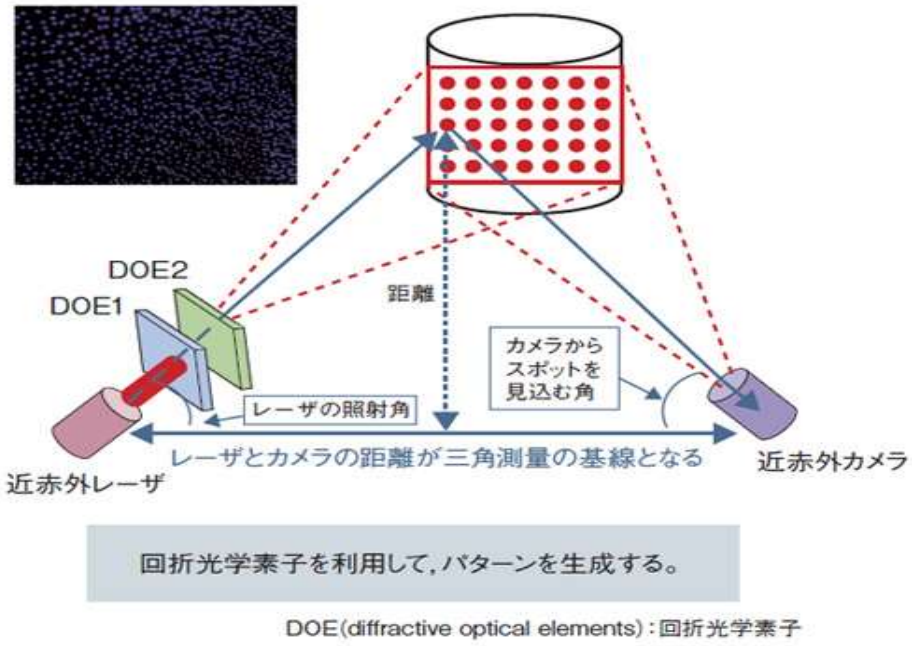


図2：三角測量を用いた距離測定

上記のように，Kinect には RGB カメラセンサと距離画像センサを搭載しており，2 種類のセンサから得た情報の整合も容易であるため，複数のセンサ情報を用いるシステムの開発[7 - 10]に向いている．

第 3 章　既存のロボットによる家事代行

3.1　概要

本章では，ロボットによる家事代行技術についての説明を示す．

3.2　Roomba

Roomba[4]とは，iRobot社が製造・販売する家庭用掃除ロボットである．直径34cm程の円盤状をしており，起動すると自動で床の掃除と充電ユニットへの帰還を行う．車輪はモータ駆動で2cm程度の段差は問題なく通過できる．接触センサや赤外線センサを搭載し，壁や家具に接触すると進行方向を変更したり，段差を感知し落下を防いだりが可能となっている．「Virtual Wall Unit」という専用の赤外線ユニットを設置することでRoombaが感知する仮想壁を作ることができ，進入禁止区域を指定できる．また，機種によっては光センサによるゴミの検知も可能であり，ゴミや汚れの多い場所を集中的に掃除することで部屋を均一にきれいにする．

図3：Roomba

3.3　食器洗い支援ロボット

単身世帯や要介護者をもつ世帯などでの食器の後片付けの負担を軽減する，家事介護支援ロボットである[11]．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おり，この中の「食器洗い」を支援するロボットである．ロボットに食器を壊さないように扱わせるには，複数のセンサ情報に瞬時に対応する必要がある．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．単純で素早い応答が求められる処理は別スレッドで並列に処理し，メインスレッドでは時間的・空間的に広い範囲の情報によって判断を行う．このように適切な動作を計画することで，複雑な作業を実現するためのソフトウェア構成法を開発している．

図4：食器洗い支援ロボット

3.4　ロボットへの直感的指示システム

これは，ユーザがマウスで手順を示すことで，小型ロボットがその通りに服たたむというものである[12]．天井に設置されたカメラセンサによって，服の認識およびロボットの位置認識を行っている．カメラセンサによって認識された服は，パソコンの画面上にモデル化されて表示される．ユーザはマウスによるドラッグアンドドロップ操作でモデル化された衣服を折りたたむことができ，その折りたたみ方に従ってロボットは自分の移動経路を決定し，折りたたみ作業を実行する．

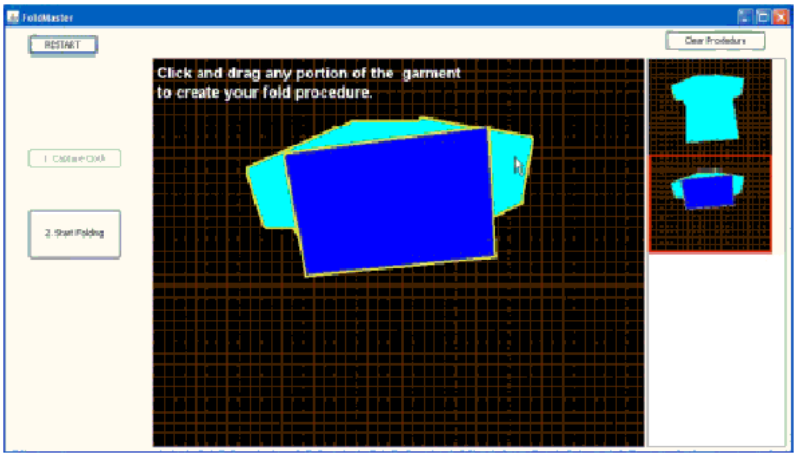


図5：モデル化された衣服の操作

服たたみロボットは4輪で服をつかむためのハンドと位置同定のためのマーカーを装備している．ハンドは，大きい衣服に対応するために幅のあるクリップタイプを採用している．ロボットが行う動作は，「前進」「後退」「旋回」「つかみ」「放し」である．たたむことが可能な衣服はTシャツ，長袖Tシャツ，タンクトップ，ズボン，靴下，タオルとなっている．

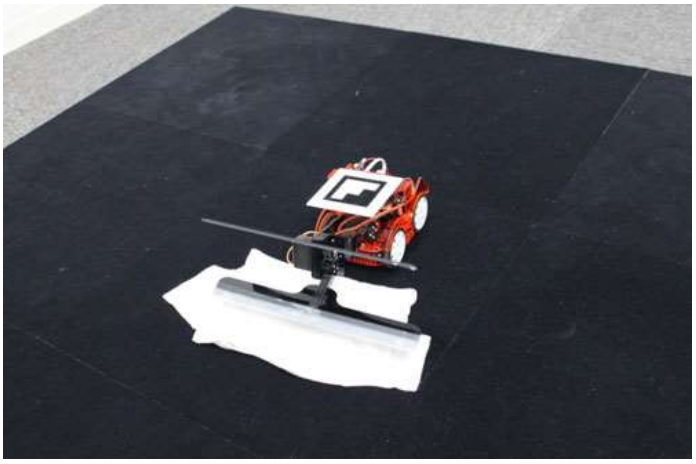


図6：ロボットの動作の様子

3.5　洗濯物たたみロボット

雑然と置かれたタオルを1枚つかみ，それを大きさに応じたたたみ方でたたむロボットである[13]．複数のセンサによって，把持物の大きさの認識や把持位置の決定，歪みの認識を行い，2本のアームによる適切なハンドリングが可能である．タオルの隣接した角を把持し，テーブルの端を擦るように用いることできれいな折り目を作っている．

図7：たたみ動作の様子

3.6　ホームアシスタントロボット

これは，複数の家事実行機能を有する家事・介護支援ロボットである[14]．実行できる家事は，「食器が乗ったトレイの運搬」「洗濯機の操作」「箒による床掃除」である．これらの作業を行うために，ロボットは家具姿勢推定と衣類発見ができるようになっている．家具姿勢推定は，距離センサとカラー画像センサを併用することで，模様の少ない家具の認識を可能としている．衣類発見は，布の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を可能としている．これにより，無造作に置かれた衣類も認識することができる．また，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できる．これには，視覚・力覚センサ情報や，計画時の姿勢と作業時の姿勢の比較を用いて作業状態を監視することで実現している．

図 8：トレイの操作

図 9：洗濯機の操作

図 10：掃除時の椅子どかし

図 11：箒による掃除

第 4 章　想定最終目標

4.1　概要

本研究の最終目標として，図 12 のような小型ロボット搭載型クローゼットシステムの開発を目指す．クローゼット上部に複数のセンサを搭載した Kinect を設置することで，洗濯物やロボットの位置を認識し，洗濯物の種類に応じた片づけ方でロボットが片づけを行う．このシステムで使用を想定するロボットは，Roomba のように地面を移動し，「つかむ」「放す」が行える小型のものとしている．

上着やズボンをハンガーに架ける形で収納することで，「服をたたむ」というロボットが苦手とする布地操作の実行回数を減らし，より短時間で正確な収納が可能となると考える．

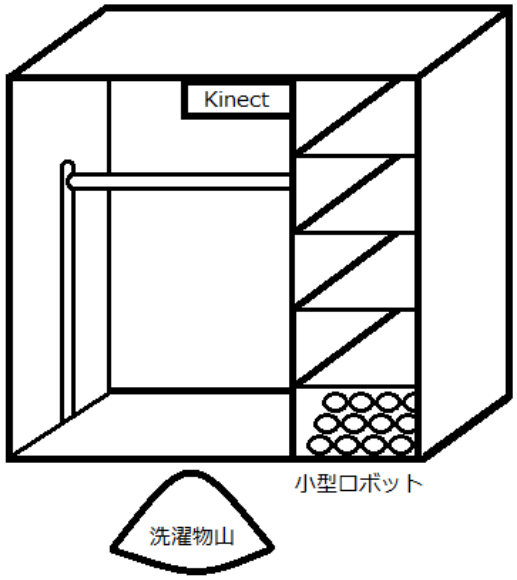


図12：目標システム

4.2　理想動作

このシステムの理想動作は次のようなものとする．

1. 対象物(洗濯物山)をシステムの範囲内に設置する．
2. 小型ロボットが洗濯物山から洗濯物 1 枚を分離させる．
3. 分離させた洗濯物の種類に応じた片づけを行う．
   3.1 上着やズボンの場合：クローゼットに収納する．
   3.2 靴下の場合：ペアリングを行いつつ空きスペースに移動させる．
4. 洗濯物山がなくなるまで 2 と 3 の動作を繰り返す．
5. 靴下をクローゼットに収納する．

6. 終了.

4.3 状態定義と実行動作

前述の動作をそのままロボットに実行させることは難しいため，洗濯物の状態を 4 つに分けて定義を行った．これによりロボットに下す命令の簡単化が可能となる．

状態 1. 複数の種類の洗濯物が一つの山になっている状態(図 13a).
状態 2. 1 つの洗濯物が山から分離している状態(図 13b).
状態 3. 靴下のペアリングが完了している状態(図 13c).
状態 4. カメラに洗濯物が映っていない状態(図 13d).

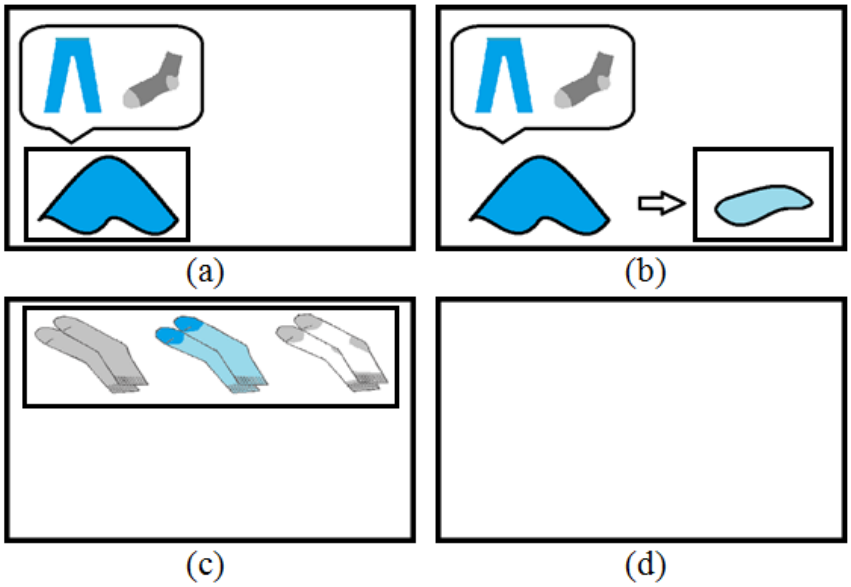


図13：洗濯物の状態

これら 4 つの状態での各実行動作と状態遷移を以下に示す. 表 2 における「対象物」は，次にどのような動作を行うかの判断をするために用いており，図 13 の矩形で囲った部分に注目している．「IF」は，対象物の状態または種類を表している．これにより，実行する処理や遷移先の状態を決定する．「THEN」は，実行する処理を表している．「状態遷移」は，次の遷移先である状態と図 14 における遷移ルートを表している．THEN の処理が実行された後に対応する状態に遷移する.

例えば現在状態 1 の場合，カメラに洗濯物山が映っているか否かを判断し，映っていれば分離処理を実行したのちに状態 2 に遷移する.

表 2：状態判定と実行動作

状態	対象物	IF	THEN	状態遷移	
1	洗濯物山	ある	分離処理実行	状態 2 へ	1-1
		ない	-	状態 3 へ	1-2
2	分離された洗濯物	ズボン	収納処理実行	状態 1 へ	2-1
		靴下	ペアリング処理実行	状態 1 へ	2-2
		ない	-	状態 1 へ	2-3
3	靴下	ある	収納処理実行	状態 3 へ	3-1
		ない	-	状態 4 へ	3-2
4	洗濯物山	ある	-	状態 1 へ	4-1
	分離された洗濯物	ある	-	状態 2 へ	4-2
	靴下	ある	-	状態 3 へ	4-3
	全洗濯物	ない	-	終了	-

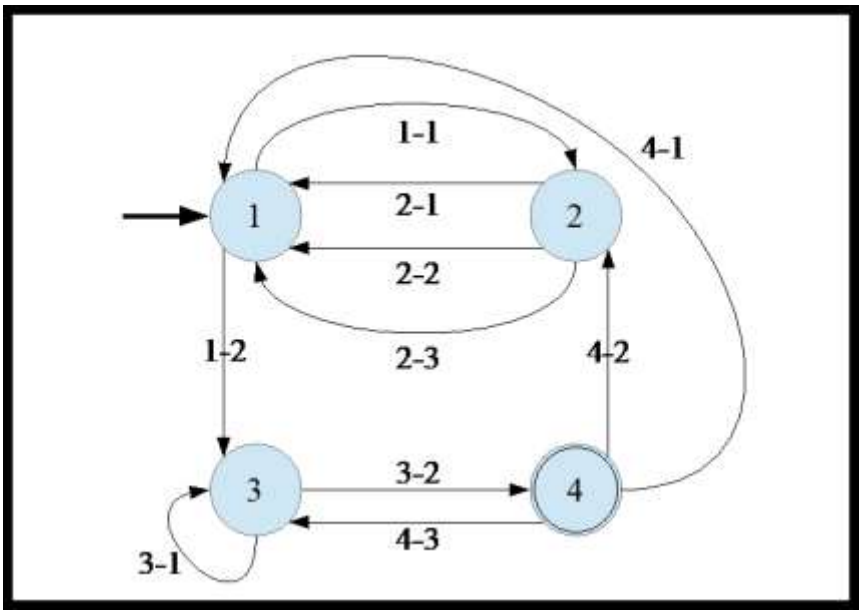


図 14：状態遷移図

第 5 章　複数センサを用いた折れ検出

5.1　概要

4章のように，システムに必要な処理は，分離処理，ペアリング処理，収納処理である．この中で，靴下はペアリングされていないと使用できない，作業に手間がかかるといった点から，靴下のペアリング処理は重要度が高いと考える．対象物である靴下の初期状態は，洗濯物山から分離した1枚の靴下がカメラセンサに映っている状態である．終了状態は，同じ柄の靴下がペアリングされて置いてある状態である．ペアリングを行う際に靴下に折れが発生している場合，靴下の柄や色を正確に認識することができない．そのためペアリングを行うためには，靴下に発生している折れを認識し修正しなければならない．

本章では，折れを検出し，折れのない状態にする処理，取得した画像から色や柄の特徴を求めペアリングする処理を実装する手法について述べる．

単一のセンサによる情報では正確な特徴を取得することは難しいため，複数のセンサによるセンサフュージョンを用いる[15][16]．今回は，赤外線距離センサによる距離画像とカラー画像カメラセンサによるRGB画像を使用する．距離画像およびRGB画像の取得は，Microsoft社のゲームコントローラであるKinect を用いて行う．これは距離画像とRGB画像の座標同期が容易に行えるためである．

5.2　折れ検出処理

この処理によって，靴下の折れを検出する．靴下に折れが発生している場合，その部分の厚さがほかの部分より2倍以上厚くなる．また，折れは曲線状ではなく直線状に発生するものである．これらのことを利用して靴下の折れを検出し，折れを修正する際に把持する位置を決定する．以下にその道程を示す．

1. カメラセンサから靴下のRGB画像の取得
2. 距離センサから靴下の深度画像を取得
3. 2枚の画像から靴下領域の切り抜き
4. 深度画像から重なり部分を検出
5. 重なり部分から高低差を検出
6. 高低差部分から直線を検出
7. 検出した直線から折れ箇所を決定
8. 検出した折れから把持位置を決定

上記の4～7の処理の詳細を以下に示す．

5.2.1　高低差の検出

折れの検出に靴下の厚みの変化を利用するため，靴下に発生している高低差の検出を行う．これには，靴下の深度画像を使用する．図15に示すように，靴下の深度画像はグレースケールで表される．このグレースケールの色は，靴下とKinectの距離に応じて変化する．今回は図16のように，床部分の輝度値を0(=黒)，靴下の輝度値を0～127に設定しており，床からの距離が遠い程靴下の色は黒に近づくようにしている．

図15：靴下の深度画像

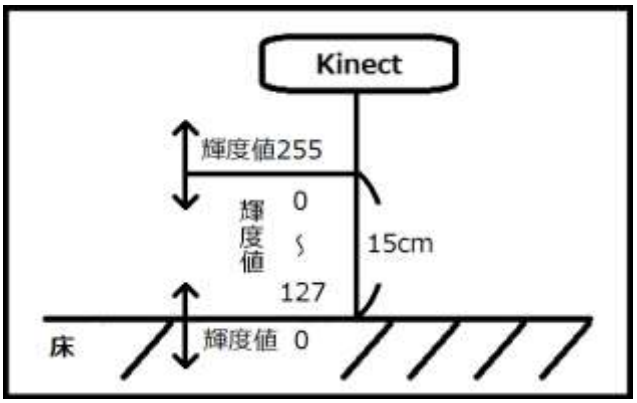


図16：認識範囲と輝度値の設定

靴下の深度画像から，他より厚くなっている部分を検出する．この検出は，輝度値が閾値より小さいピクセルを切り出すことで行う．閾値は靴下全体の輝度値から求めた平均値を用いる．切り出した部分に対して輪郭を抽出することで，靴下に発生している高低差部分を取得することができる．図15の靴下に対して切り出し処理を行ったものが図17，輪郭抽出処理を行ったものが図18となる．

図17：厚い部分の画像

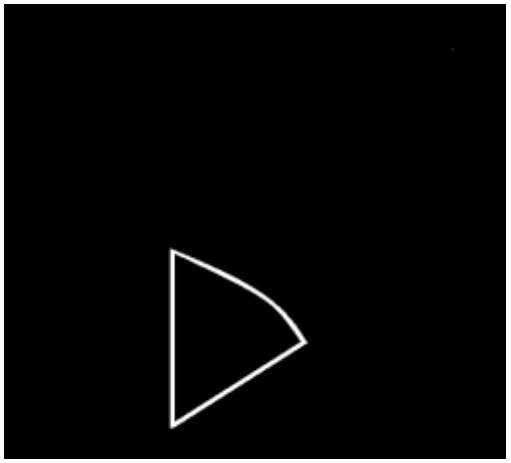


図18：高低差画像

5.2.2　折れ箇所の決定

5.3.1で検出した高低差画像に対して直線検出を行う．図18の画像に対してこの処理を行ったものが図19である．検出された直線を折れ候補として保存しておき，靴下の外周部画像との比較を行う．外周部画像は靴下の深度画像の輪郭を抽出したものである(図20)．外周部と一致する場所に存在する折れ候補の直線を，靴下に発生している折れ箇所に決定する．　図15の靴下に対して検出された折れ箇所を図21に示す．

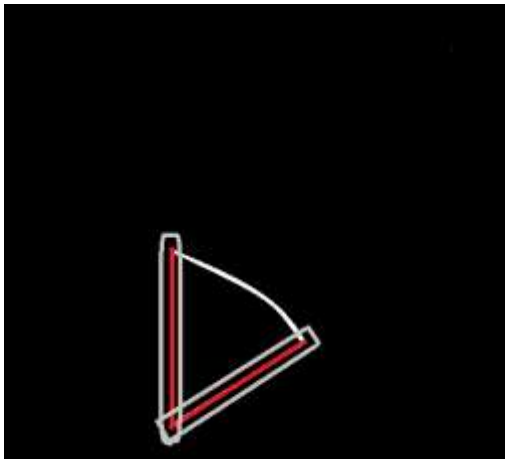


図19：折れ候補

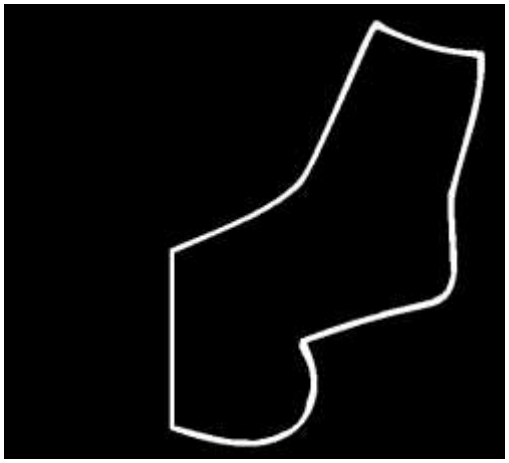


図20：靴下の外周部

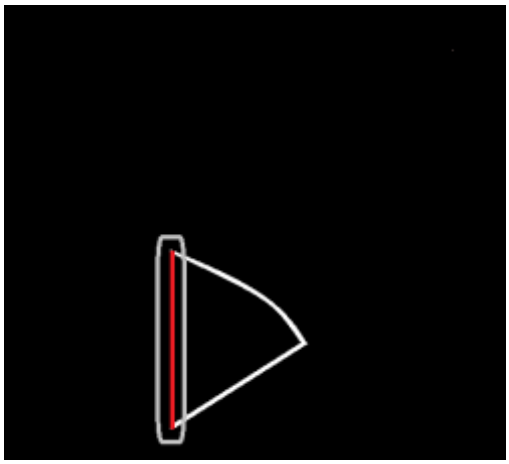


図21：決定した折れ箇所

5.2.3　把持位置の決定

決定した折れ箇所から，折れ修正時に用いる把持位置の決定を行う．図22の線分ABは5.3.2で決定した折れであり，線分ABの中点を点Cとする．点Cを通り線分ABに対して垂直な直線を引き，高低差部と交差した点を点Dとする．この点Dが，折れ修正時の把持位置であり，把持位置を点Cに対して点対称の位置に移動させることで折れを修正するものとする．

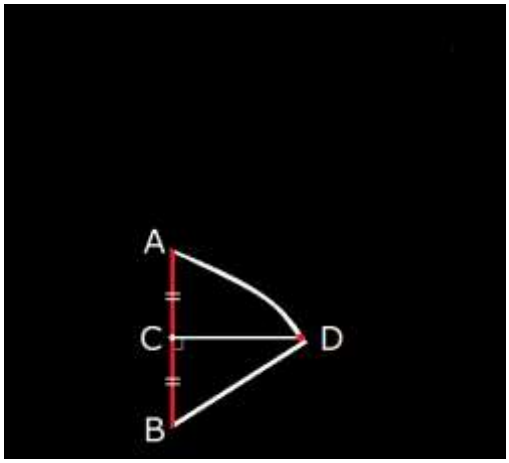


図22：把持位置

5.3　ペアリング処理

折れを修正した靴下のRGB画像から，RGB各色の靴下に対する割合を求め，特徴量として用いることで同じ靴下であると決定する．

図23：サンプルイラスト①

図23の靴下を例にすると，図24～図26の結果を得ることができる．この結果を記憶しておくことで，新たに入力された靴下の結果と比較し，既知であるかの判断を行うことができる．

図24～図26の縦軸は靴下に対する各色の割合，横軸は各色の輝度値を示している．

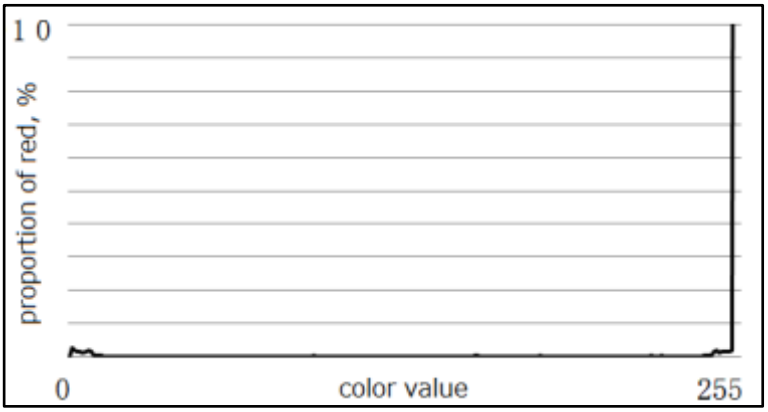


図24：REDの輝度値とその割合

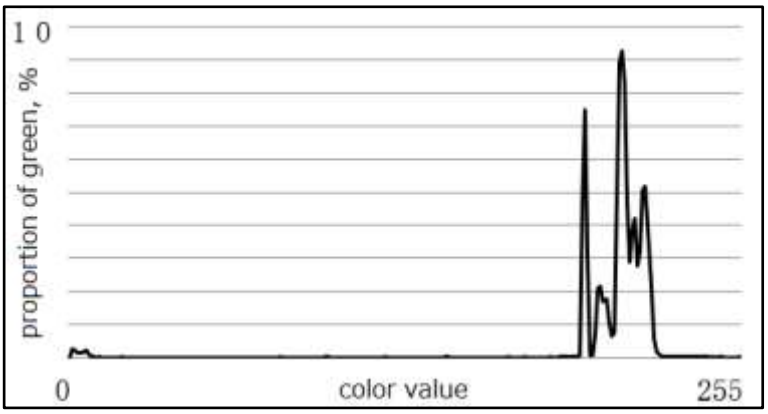


図25：GREENの輝度値とその割合

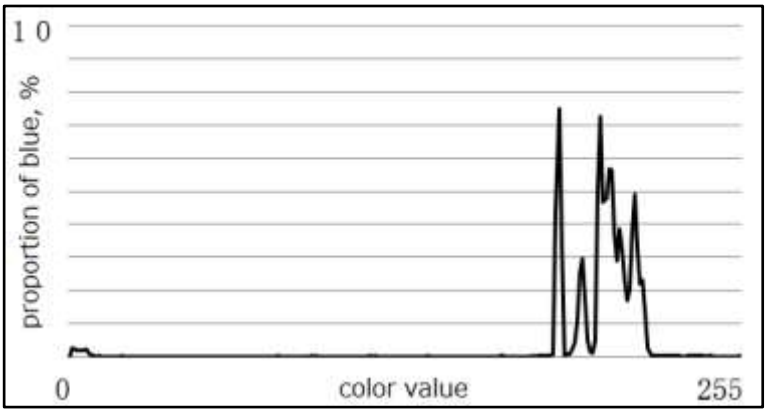


図26：BLUEの輝度値とその割合

現在入力されている靴下が，記憶している靴下のペアであると判断した場合は，ペア同士に同じ番号を割り当てる．ペアとなる靴下をまだ検出していない靴下には異なる番号を割り当てる．この処理を繰り返して靴下のペアリングを完了する．

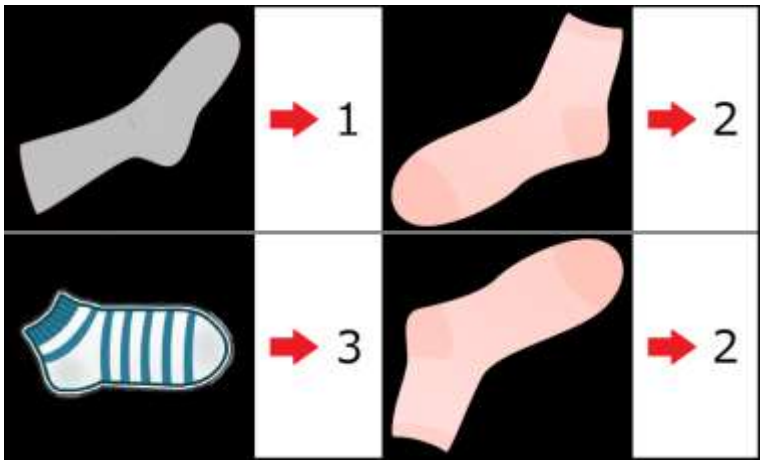


図27：マッチング一例

5.4 開発環境

今回の開発環境を以下に示す.

表3：開発環境

OS	Windows 7 Home Premium 64bit
CPU	Intel® Core™ i5-470UM 1.33GHz/2コア
メモリ	8GB
開発ツール	Microsoft Visual Studio 2010
開発言語	C++
ライブラリ	OpenNI 1.5.2
使用デバイス	Kinect v1

OpenNIとはKinectのセンサ部を開発したPrimeSence社が中心となって開発したオープンソースのライブラリである．以下にOpenNIの仕様を示す.

表4：OpenNIの仕様

対応OS	WindowsXP, Vista, 7 Ubuntu Mac OS X
開発言語	C, C++, C#
RGBカメラ解像度	640×480
距離カメラ解像度	640×480
認識距離範囲	500-10000mm

次にKinect v1のスペックについて以下に示す.

表5：Kinect v1の仕様

対応OS	WindowsXP, Vista, 7 Ubuntu Mac OS X
接続	USB2.0
RGBカメラ解像度	RGB_640x480(30fps), RGB_1280x960(12fps)
距離カメラ解像度	80x60(30fps), 320x240(30fps), 640x480(30fps)
RGB視野	(水平) 62度, (垂直) 48.6度
深度視野	(水平) 57度, (垂直) 43度

第 6 章　実験と評価

6.1　概要

本章では，5章で提案した処理によって，折れの発生位置の判定と，折れ修正後のペアリングが正しく行われるかの実験について示す．

6.2　実験手順

今回行った実験の手順について示す．なお，靴下の移動などの操作は，ロボットではなく人の手で行っている．

6.2.1　初期配置

Kinectセンサに靴下を認識させるために図28のようにKinectと靴下を配置した．Kinectは床から約2mの高さの位置にカメラセンサが下を向くように固定している．靴下は乱雑に置かれた状態のものをKinectの下まで引っ張って配置している．

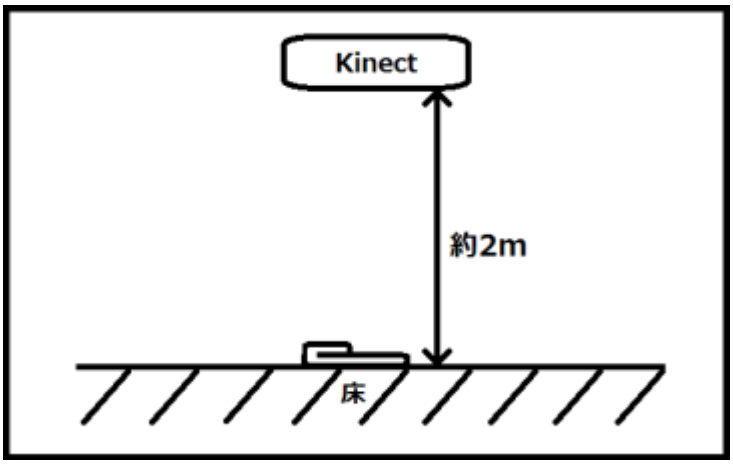


図28：初期配置

6.2.2　折れ検出実行

認識した靴下に対して折れ検出を実行した．折れが発生している場合は，折れの位置の表示と指定された把持位置の移動，発生していない場合はその旨の表示を行った．1枚の靴下に対して，折れが無いと判断されるまで処理を行い，これを用意したすべての靴下に対して実行した．本来の機能としては必要ではないが，今回の実験の結果をわかりやすくするために，5.3で示したような靴下の画像を出力するようにしている．

6.2.3　折れ検出性能評価

6.2.2の結果から折れ検出処理の性能を，情報検索における性能評価指数の１つである適合率･再現率･F値で評価した．この3つの値は，最大値1，最小値0でありF値が1に近い程性能が良いことを意味している．求め方を以下に示す．

$$\text{適合率} = \frac{\text{システムが折れていると判断した靴下の中で実際に折れている靴下の総数}}{\text{システムが折れていると判断した靴下の総数}}$$

$$\text{再現率} = \frac{\text{システムが折れていると判断した靴下の中で実際に折れている靴下の総数}}{\text{用意した折れている靴下の総数}}$$

$$\text{F値} = \frac{2 \times \text{適合率} \times \text{再現率}}{\text{適合率} + \text{再現率}}$$

6.2.4　ペアリング実行

折れ検出処理を実行した靴下に対してペアリング処理を行った．新しく認識された靴下ならば新しい数字を，既知の靴下であれば同じ数字を割り当てる．システムが作成したペアの数，正しくペアリングされた数を記録し性能評価に用いる．

6.2.5　ペアリング性能評価

6.2.4の結果からペアリング処理の性能を，適合率･再現率･F値で評価した．求め方を以下に示す．

$$\text{適合率} = \frac{\text{システムが作成したペアの中で正しいペアの総数}}{\text{システムが作成したペアの総数}}$$

$$\text{再現率} = \frac{\text{システムが作成したペアの中で正しいペアの総数}}{\text{用意したペアの総数}}$$

$$\text{F値} = \frac{2 \times \text{適合率} \times \text{再現率}}{\text{適合率} + \text{再現率}}$$

6.3　実験対象

実験は表6に示した対象に行った．実験対象の靴下の画像を図28に示す．

表6：実験対象

用意した靴下数	30枚
用意したペア数	15組
測定回数	10回

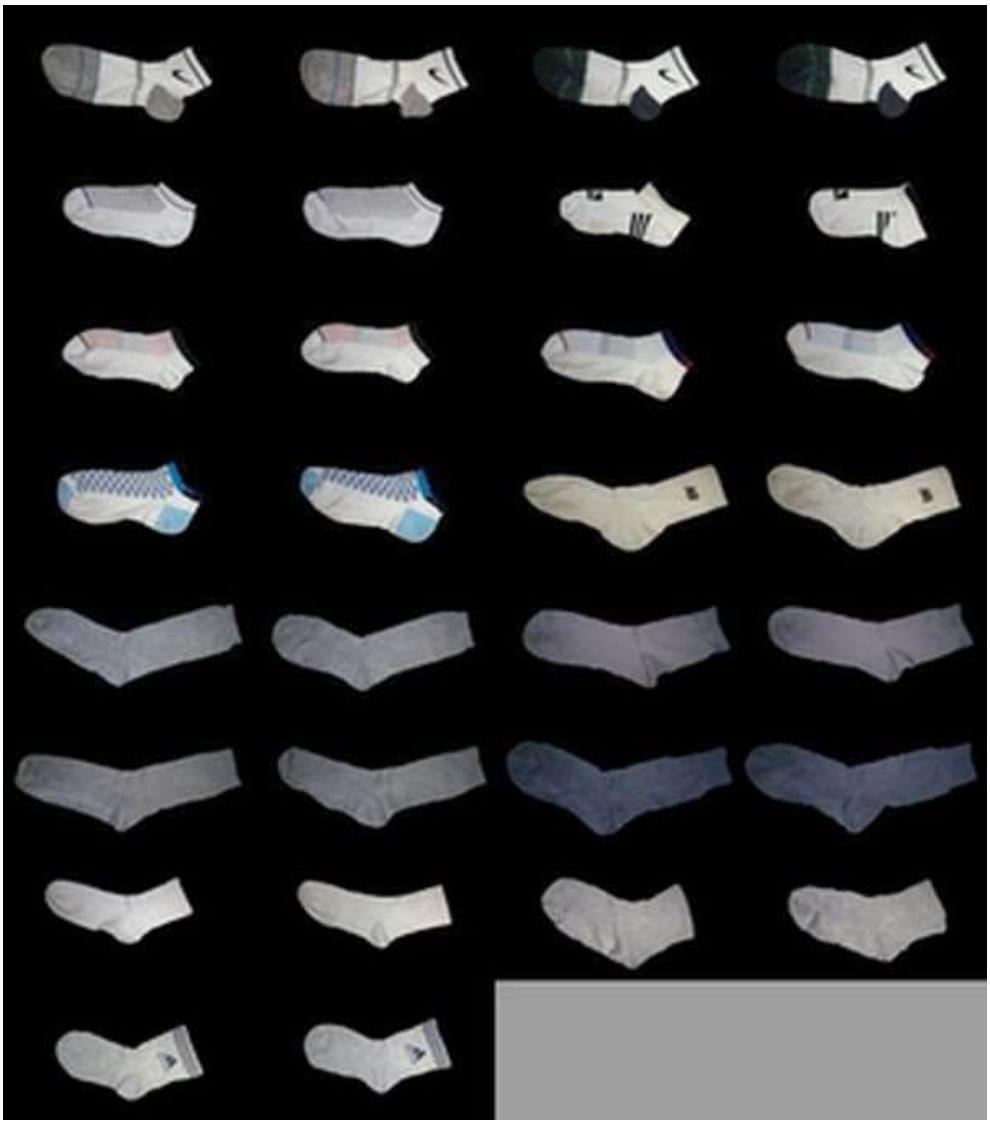


図28：実験対象の画像

6.4　折れ検出実験

6.4.1　実験目的

本研究で構築した折れ検出処理が，どの程度の性能を持っているかの検証のために折れ検出実験を行った.

6.4.2　実験結果

提案した折れ検出手法による折れ発生位置検出実験の結果の一例を以下に示す.

図30：RGB画像

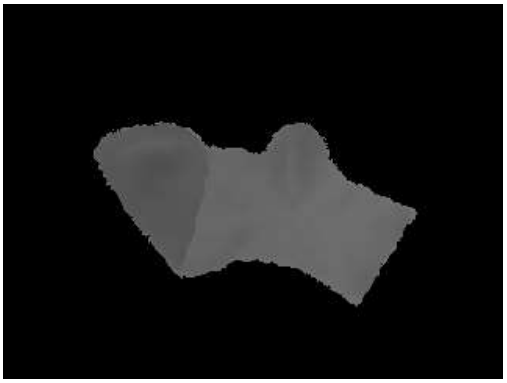


図31：取得深度画像

図32：折れ部画像

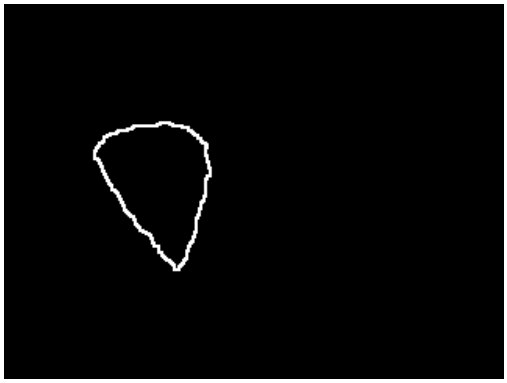


図33：高低差画像

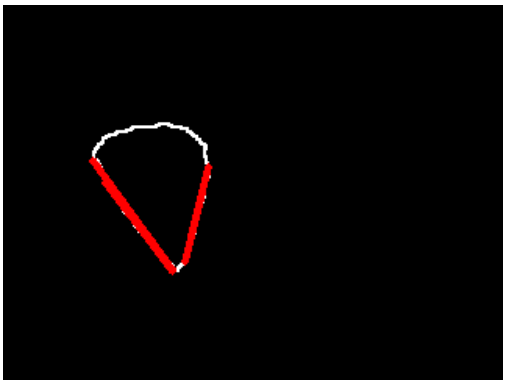


図34：折れ候補

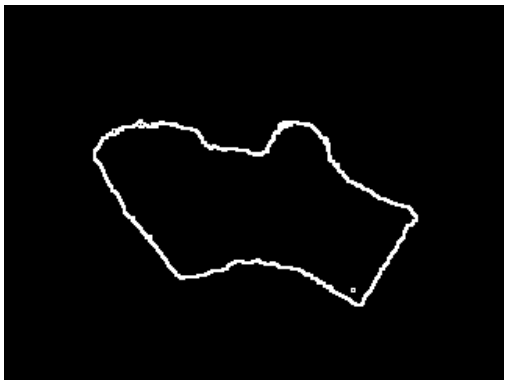


図35：外周部画像

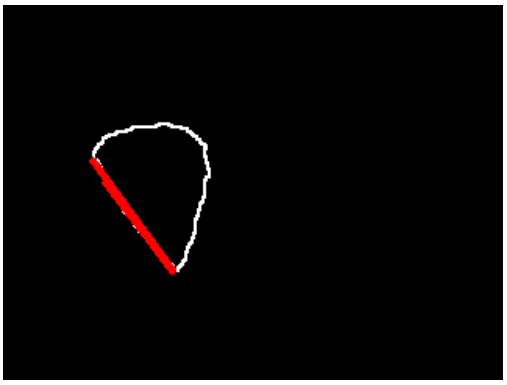


図36：折れ箇所

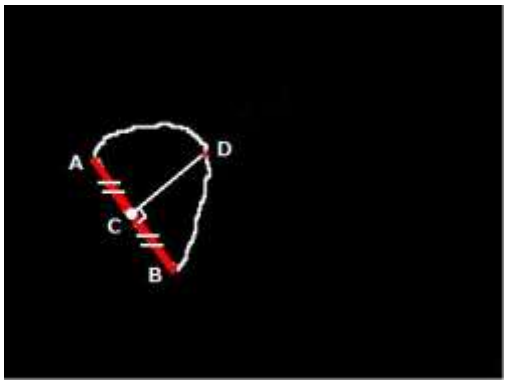


図37：把持位置

本例に使用した靴下は図30のものである．本例における閾値は図31より98.7となった．この閾値より輝度値が小さい「厚い部分」を表す画像が図32である．この画像に対し輪郭抽出を行ったものが，図33の高低差画像である．この靴下の折れ候補は，図34のように2か所挙げられた．図34と図35の比較によって図36の折れ箇所が決定された．この靴下における把持位置は，図37より点Dに決定された．

また，本実験によって性能評価は表7のようになった．

表7：折れ検出性能評価

適合率	再現率	F値
0.83	0.75	0.79

6.4.3　考察

表7より適合率が0.83と高い値を示した．これは，折れが発生しているとシステムが判断した靴下の83%が実際に折れていたことを示している．これにより，本システムの折れ検出結果の信頼度は高いといえる．また，再現率は0.75となり，低くない値ではあるが，適合率と比較すると低い．これは，薄手の生地でできた靴下やストッキングのように折れを正確に検出できないものが存在していたためである．折れ検出の精度を向上させるためには，これらのものに対する策が求められる．

6.5　ペアリング実験

6.5.1　実験目的

システム全体の性能を評価するために，折れ検出処理を施した靴下に対するペアリング実験を行った.

6.5.2　実験結果

ペアリング実験の結果の一例を表 8～表 11 に示す.

表8：ペアリング実験結果(1)

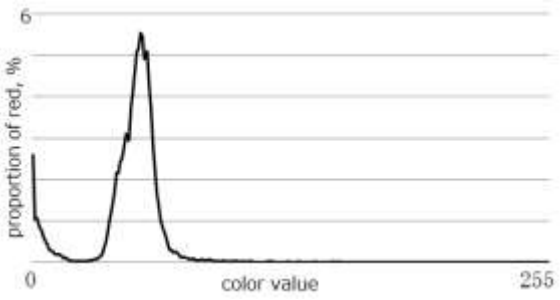
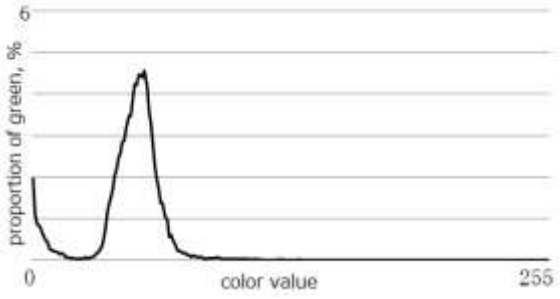
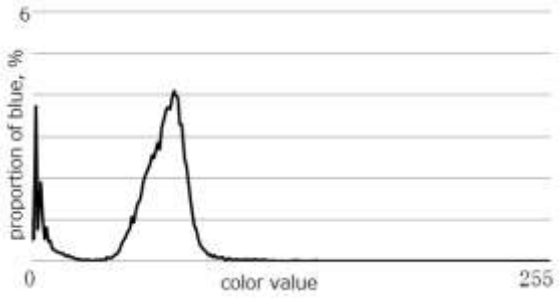
靴下画像	カラーグラフ(Red, Green, Blue)	出力
	  	1

表9：ペアリング実験結果(2)

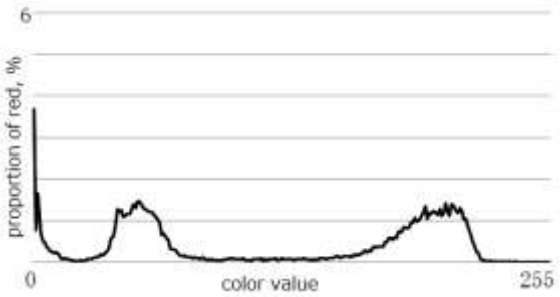
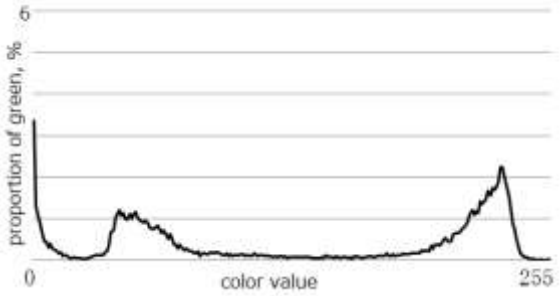
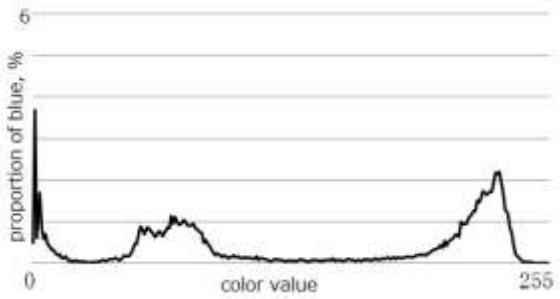
靴下画像	カラーグラフ(Red, Green, Blue)	出力
	  	2

表10：ペアリング実験結果(3)

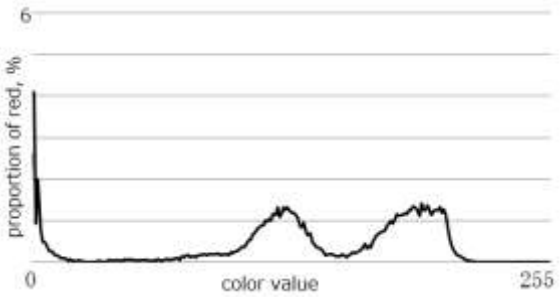
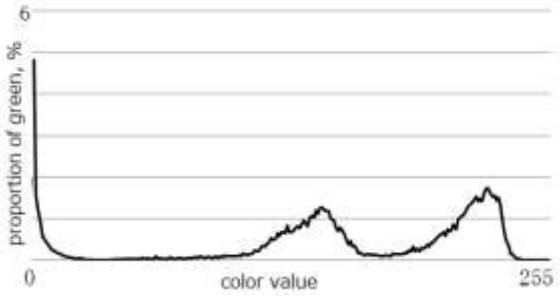
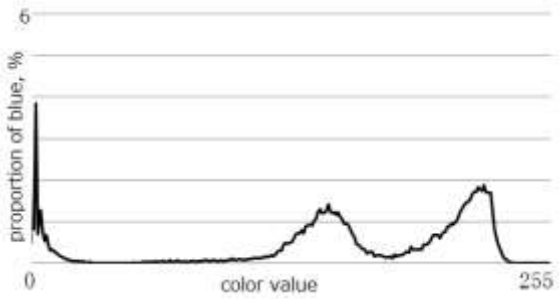
靴下画像	カラーグラフ(Red, Green, Blue)	出力
	  	3

表11：ペアリング実験結果(4)

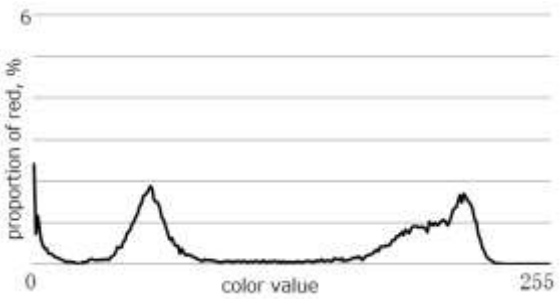
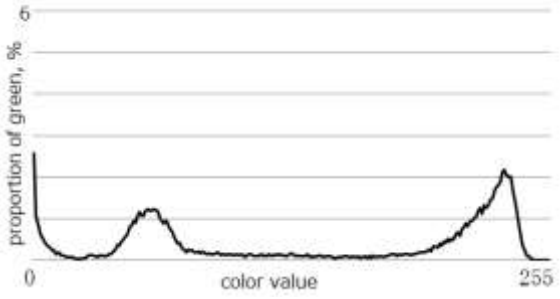
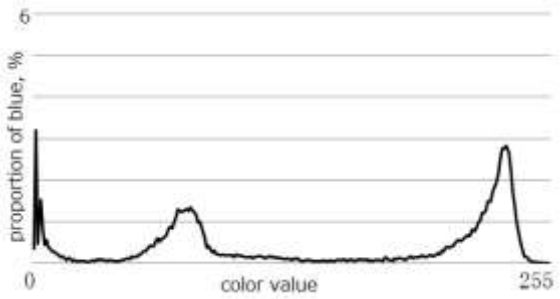
靴下画像	カラーグラフ(Red, Green, Blue)	出力
	  	2

表 8～表 11 の各 3 つのグラフは，靴下に対する各色(上から赤，緑，青)の割合を示している．グラフの形状が似ている靴下には同じ出力がされている．

また，以上の結果から求めたペアリング性能の評価は表12のようになった．

表12：ペアリング性能評価

適合率	再現率	F値
0.69	0.73	0.71

6.5.3　考察

以上の実験より適合率は0.69という値を示した．これは，システムがペアであるとした靴下の中の69%が正しいペアであることを表している．また，再現率は0.73という値を示した．これはシステムが，対象となる靴下ペアの73%をペアとして認識できるということを表している．

適合率があまり高くない理由として，ペアリングの指標が1つであるため，1つの靴下に対して正しくないペア候補が複数出現し，適合率を求める際の分母である「システムが発見したペア」の値が大きくなってしまうためと考えられる．

これらの値を向上するためには，ペアリングに使用する指標の数を増やすなどが考えられる．

第 7 章　画像分割を用いたペアリング

7.1　概要

5章では，靴下全体に対するRGB各色の割合を比較することで，ペアリングを行う手法とした．しかし，ペア候補の複数出現や，偶然色の割合が似た靴下があった場合に，正確なペアリングを行うことができない．そこで我々は，つま先部分およびロゴム部分に靴下の特徴が出ると考え，靴下をペアリングする際の画像を分割することでペアリング精度の向上を図る手法を提案する．

7.2　画像分割方法

折れを修正し取得した靴下の画像を3分割し，両端の画像のRGB各色の割合によってペアリングを行う．例として図38の靴下画像を分割すると図39，図40の画像を得ることができる．

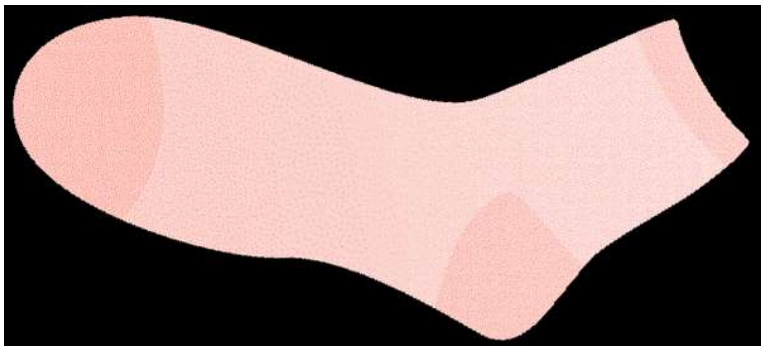


図38：サンプルイラスト②

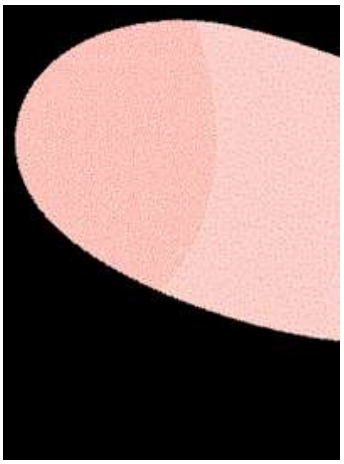


図39：分割画像(指先部)

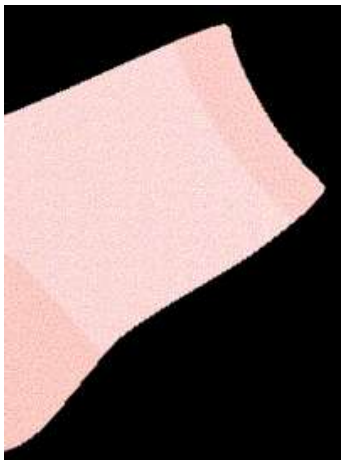


図40：分割画像(ロゴム部)

この分割した図39，図40の画像に対してペアリング処理を行うとそれぞれ以下の結果を得ることができる．

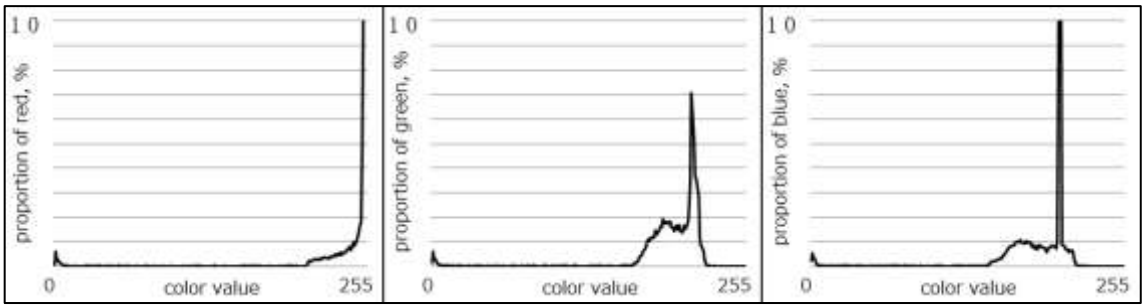


図41：分割画像①の各色輝度値とその割合

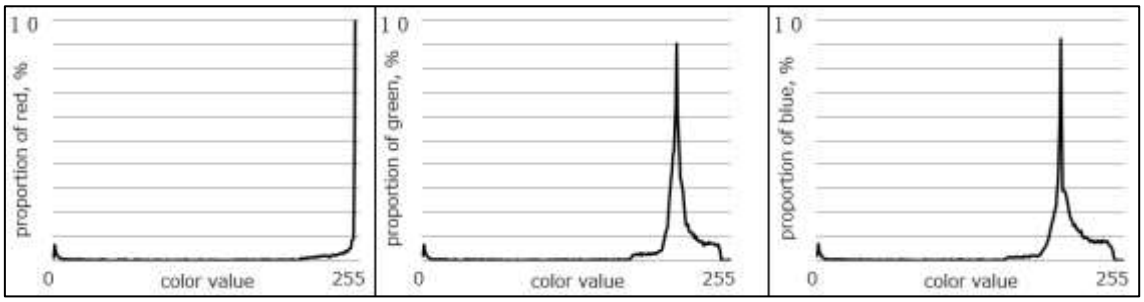


図42：分割画像②の各色輝度値とその割合

この2つの指標を用いてペアリングを行うことで，1つの指標では困難だったワンポイントの違いも識別できるようになる．

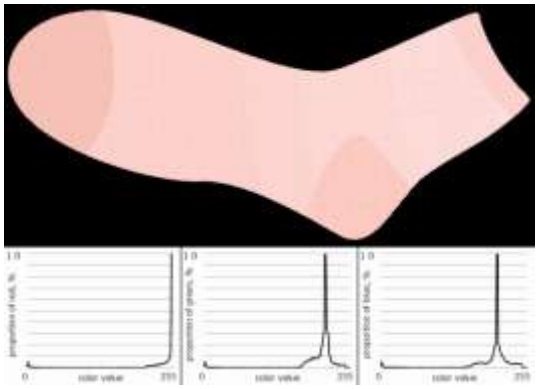


図43：サンプル靴下(1)

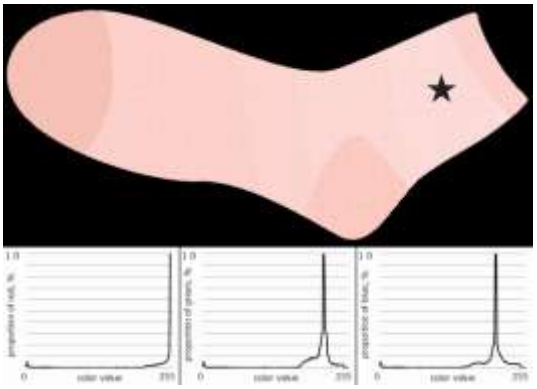


図44：サンプル靴下(2)

図43の靴下と図44の靴下を比較した場合，図44の靴下のような，ロゴム部分にワンポイントがあるものは，靴下全体を使ったペアリングでは違いが出ず，見分けることができない．しかし，指先部とロゴム部に分割してペアリングを行うと，図45のようにロゴム部の結果に違いが出る．これによって，分割なしのペアリングより精度の良いペアリングが行える．

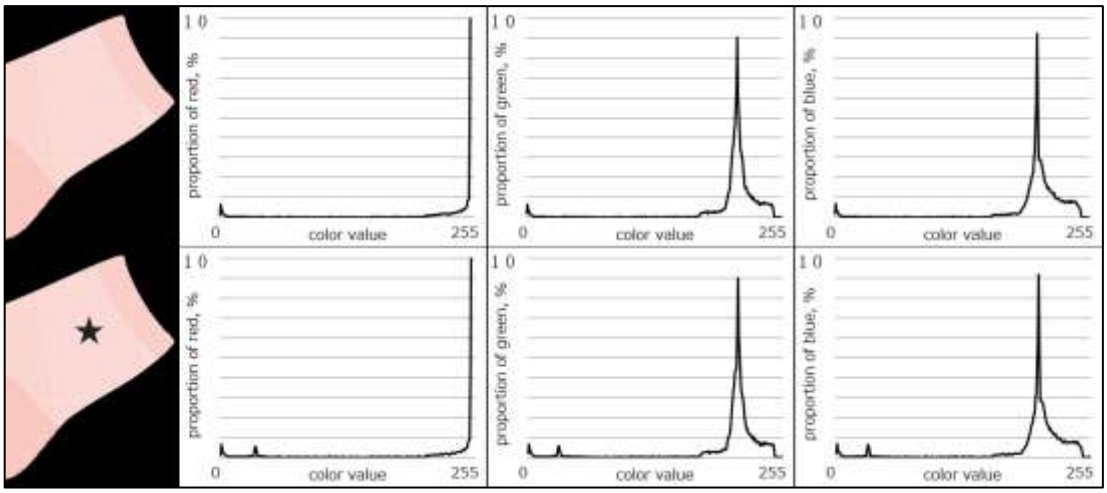


図45：ロゴム部の結果比較

7.3 開発環境

今回の開発環境は5.5で記したものと同じである.

第 8 章　実験と評価

8.1　概要

本章では，7章で提案した手法によるペアリングの評価実験について示す．

8.2　実験手順

今回行った実験の手順について示す．

8.2.1　初期配置

Kinect センサや靴下は 6.2.1 と配置で実験を行った．Kinect は床から約 2m の高さの位置にカメラセンサが下を向くように固定している．

8.2.2　ペアリング実行

画像分割を用いたペアリング処理を実験対象となる靴下に行った．新しく認識された靴下ならば新しい数字を，既知の靴下であれば同じ数字を割り当てる．システムが作成したペアの数，正しくペアリングされた数を記録し性能評価に用いる．

8.2.3　ペアリング性能評価

8.2.2の結果からペアリング処理の性能を，適合率･再現率･F値で評価した．求め方を以下に示す．

$$\text{適合率} = \frac{\text{システムが作成したペアの中で正しいペアの総数}}{\text{システムが作成したペアの総数}}$$

$$\text{再現率} = \frac{\text{システムが作成したペアの中で正しいペアの総数}}{\text{用意したペアの総数}}$$

$$\text{F値} = \frac{2 \times \text{適合率} \times \text{再現率}}{\text{適合率} + \text{再現率}}$$

8.3 実験対象

実験は表13に示した対象に行った．実験対象の靴下の画像を図46に示す．

表13：実験対象

用意した靴下数	30枚
用意したペア数	15組
測定回数	10回

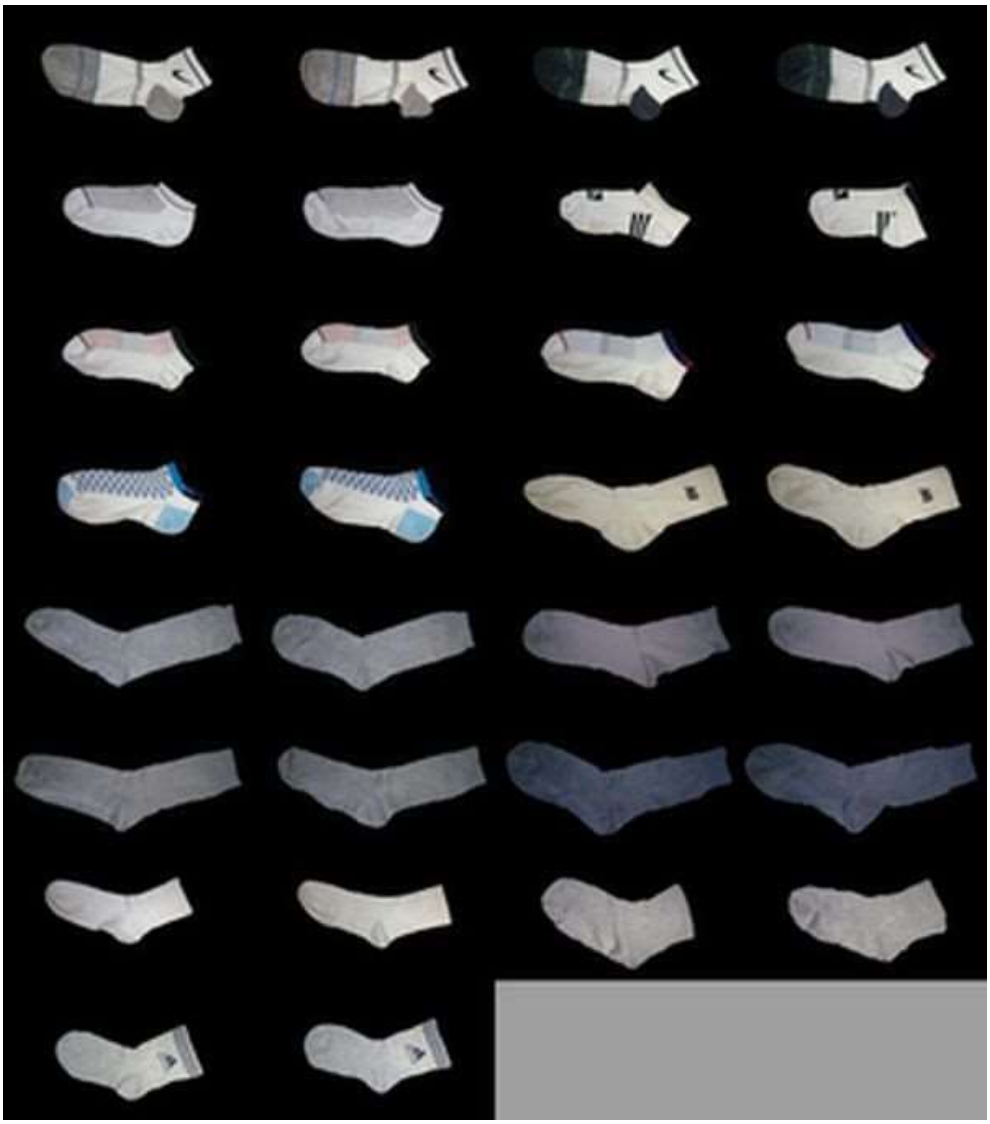


図 46：実験対象の画像

8.4　ペアリング実験結果

画像分割を用いたペアリング実験の結果の一例を表 13～表 20 に示す.

表13：実験靴下一例(1)

靴下全体画像	靴下指先画像	靴下ロゴム画像
		

表14：結果グラフ一例(1)

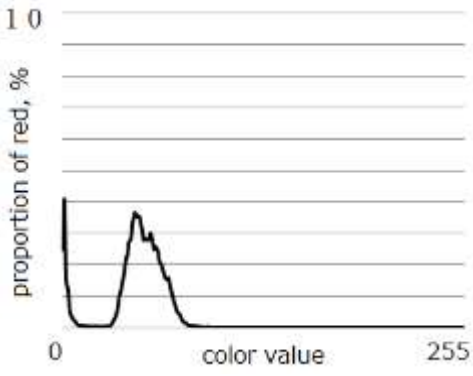
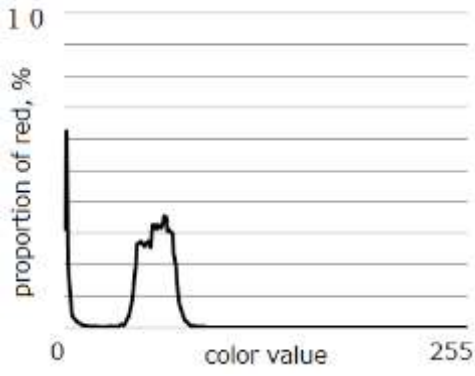
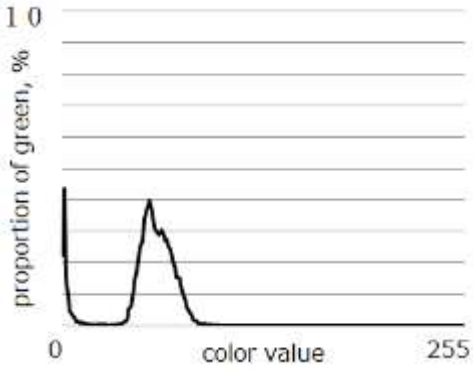
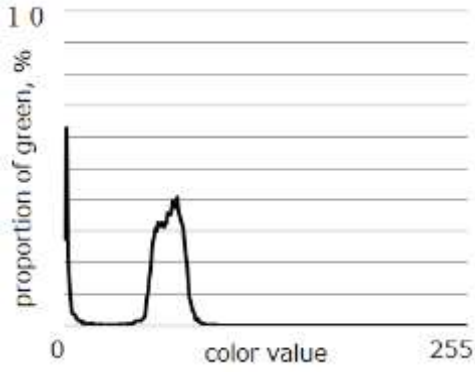
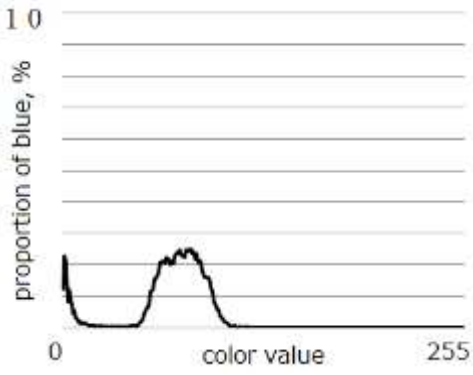
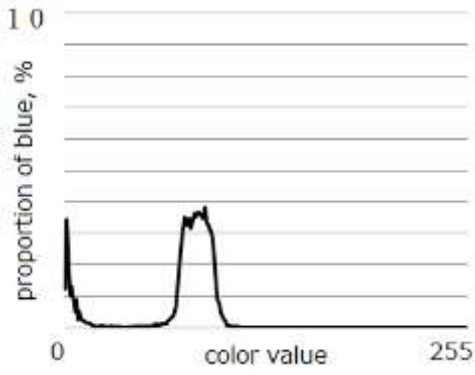
指先部グラフ(Red, Green, Blue)	ロゴム部グラフ(Red, Green, Blue)	出力
		1
		
		

表15：実験靴下一例(2)

靴下全体画像	靴下指先画像	靴下ロゴム画像
		

表16：結果グラフ一例(2)

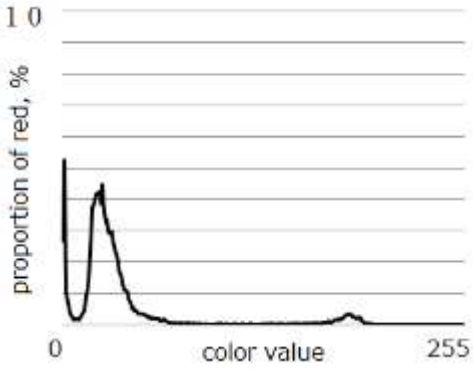
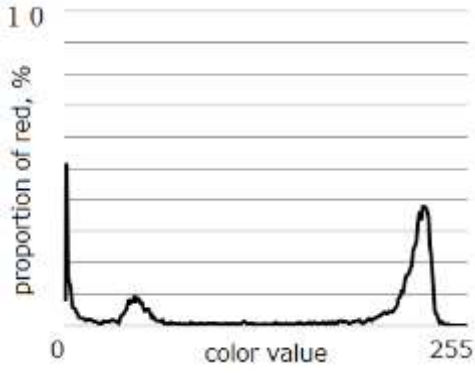
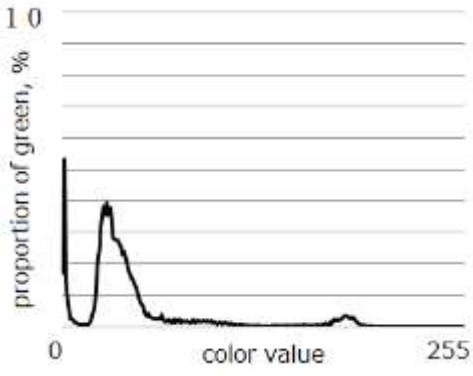
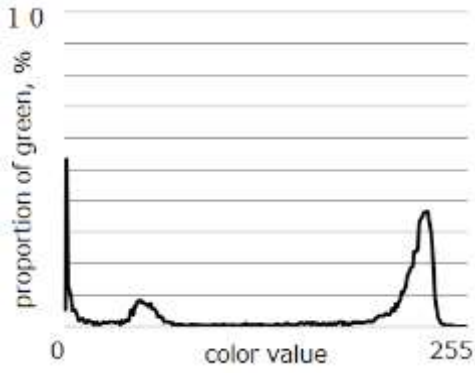
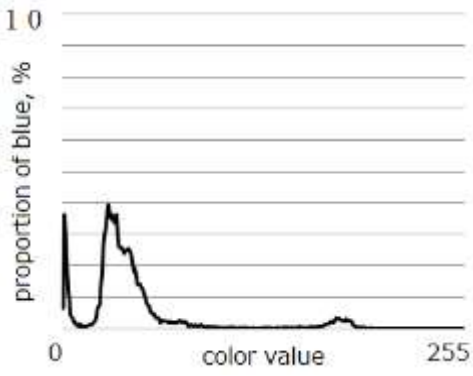
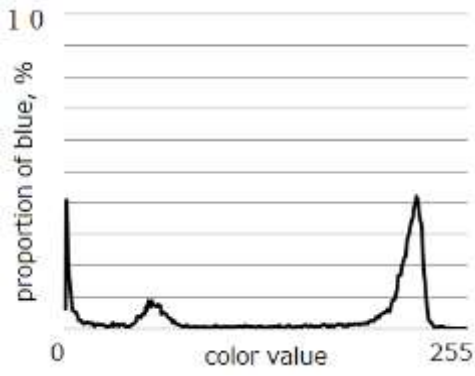
指先部グラフ(Red, Green, Blue)	ロゴム部グラフ(Red, Green, Blue)	出力
		2
		
		

表17：実験靴下一例(3)

靴下全体画像	靴下指先画像	靴下ロゴム画像
		

表18：結果グラフ一例(3)

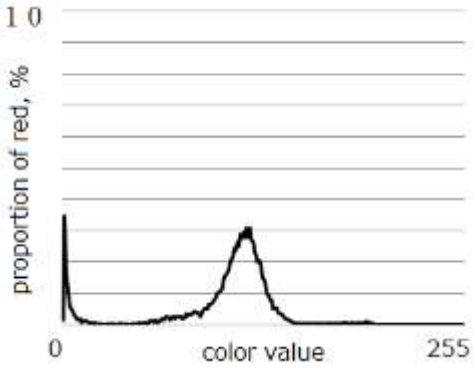
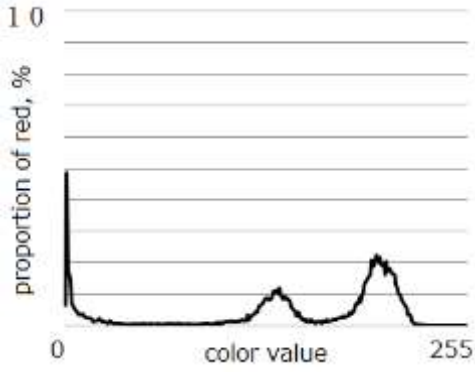
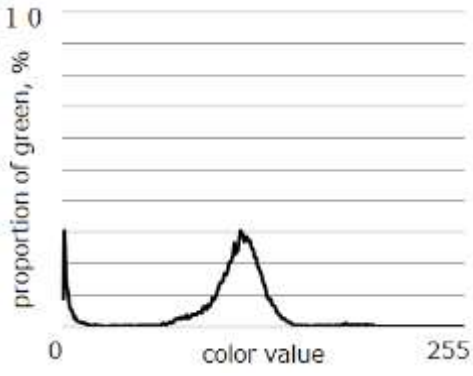
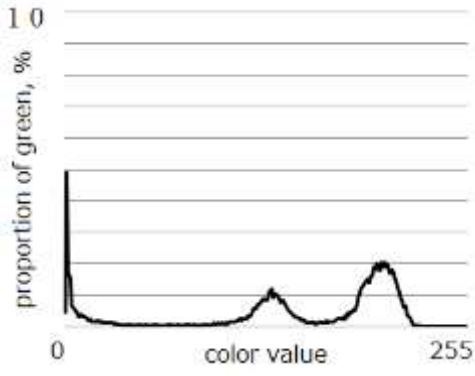
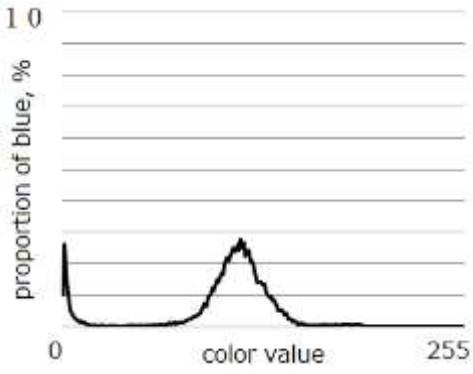
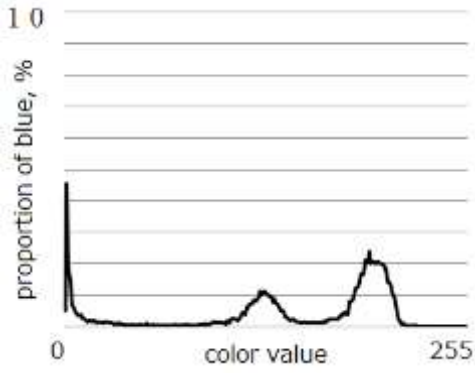
指先部グラフ(Red, Green, Blue)	ロゴム部グラフ(Red, Green, Blue)	出力
		3
		
		

表19：実験靴下一例(4)

全体画像	指先画像	ロゴム画像
		

表20：結果グラフ一例(4)

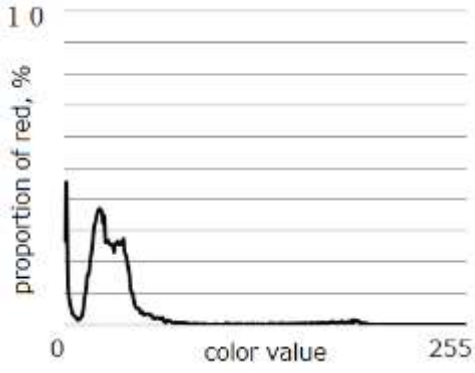
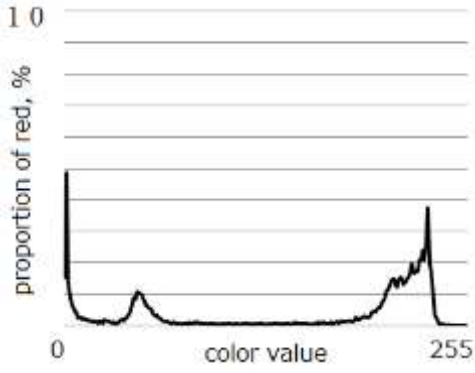
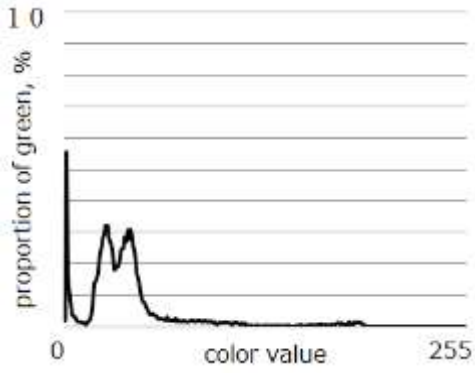
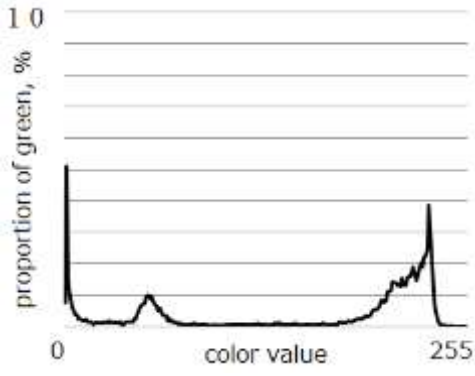
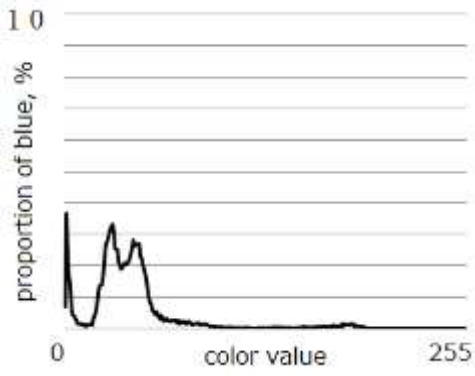
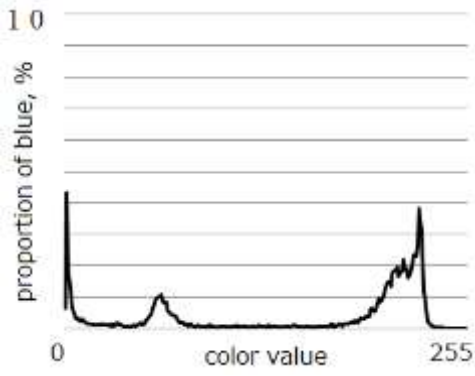
指先部グラフ(Red, Green, Blue)	ロゴム部グラフ(Red, Green, Blue)	出力
		2
		
		

表13，表15，表17，表19は，実験時の靴下の全体画像，指先部画像，ロゴム部画像である．表14，表16，表18，表20は，指先部およびロゴム部に対する各色(上から赤，緑，青)の割合を示している．

また，以上の結果から求めた画像分割ペアリングの性能評価は表21のようになった．

表21：画像分割ペアリング性能評価

適合率	再現率	F値
0.81	0.87	0.84

8.4.1 考察

以上の実験より画像分割ペアリング手法の適合率は 0.81 という値を示した．6.5.2 の表 12 と比較すると 12 ポイントの向上が見られる．この要因として，指標を 2 つにしたことで，靴下 1 枚に対してのペア候補数が 1 に近づいたことが考えられる．つまりシステムが認識したペア数が用意したペア数に近い値をとるようになったということである．よって，画像を分割することでペアリング候補を絞ることができた．

第 9 章　まとめ

本論文では，ロボットに洗濯物の片づけを行わせるシステムの一環として，複数センサを用いることで靴下のペアリングを行う手法を提案した．ペアリングを行うために，靴下の厚さの違いを利用した折れ検出処理と，靴下に対する色の割合を利用したペアリング処理の2つの処理を実装した．提案手法の性能評価として，折れ検出処理とペアリング処理の実験を行った．実験の結果，実験対象である靴下の8割近くの折れを検出することができた．しかし，ペアリングに関して，1つの靴下に対して複数のペア候補が検知され，結果適合率が0.70を下回ることとなった．原因としてペアリングに用いた指標が1つであるために，ある程度似ている靴下はペア候補になったと考えられた．

そこで，指標を2つ使用したペアリング手法として，靴下画像を分割し指先部と口ゴム部の2つを用いる手法を提案した．この手法の評価を行うために，同じ条件でペアリング実験を行った．実験の結果，指標が1つのペアリング手法より，12ポイントの評価値の向上が見られた．1つの靴下に対するペア候補も1に近いものとなり，本手法は有効であることが判明した．

今後の課題として，折れ認識を靴下だけでなく上着やズボンにも対応できるように，認識範囲や厚さに対しての輝度値の変化の調整を行うといったことが挙げられる．また，今回の実験において人力で行った操作をロボットに行わせるために，ロボットと同期させた処理の実行ができるようにする必要がある．

謝辞

本論文を作成するに当たり，ご指導頂いた三好力教授に感謝いたします．また，様々な助言を下さった三好研究室の皆様や，多忙であるにも関わらず実験データの集計に協力してくださった皆様に心から感謝したします．まことにありがとうございました．

参考文献

[1] Okada,K.,Ogura,T.,Haneda,A., Fujimoto,J., Gravot,F. and Inaba,M., Humanoid motion generation system on HRP2-JSK for daily life environment, In Proc. ICMA 2005, IEEE(2005), 4, 1772-1777.

[2] 「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>

[3] 「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>

[4] “iRobot Corporation” <http://www.irobot.com/>

[5] “ECOVACS” <http://www.ecovacs-japan.com/>

[6] “Kinect – Xbox.com” <http://www.xbox.com/ja-JP/kinect>

[7] 「Kinectの技術を採用した障害者支援サービス”OAK” キッザニア東京で体験会実施」 <http://www.famitsu.com/news/201210/05022394.html>

[8] “wi-Goプロジェクト” <http://vimeo.com/24542706>

[9] 「Kinectによる側弯症計測システム」<http://k4wa.com/report/grandprix.html>

[10] 「三井ホーム、KinectでTV/ブラインド/照明が操作できる次世代住宅公開」 <http://pc.watch.impress.co.jp/docs/news/20120911_558720.html>

[11] 「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>

[12] 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)

[13] “Willow Garage” <https://www.willowgarage.com/>

[14] 「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>

[15] 中村薫(2011)『Kinectセンサープログラミング』秀和システム.

[16] Fan Liu, Jinhui Tang, Ruizhen Zhao and Zhenmin Tang, Abnormal behavior recognition system for ATM monitoring by RGB-D camera, Proceedings of the 20th ACM international conference on Multimedia, (2012), 1295.

[17] E.Torgerson and F.W.Paul, Vision-guided robotic fabric manipulation for apparel manufacturing, Proceedings of the 1987 IEEE international conference on Robotics and Automation, 2, (1987), 1196.

[18] Paul.M.Taylor, Sensory Robotics for the Handling of Limp Materials, Springer-Verlag, (1990), 141.

[19] 濱島京子，柿倉正義，”布地物体展開手順のプランニング - 塊状洗濯物の分離 - ”，日本機械学会論文集C編，Vol.63，No.607，pp.333-340，1997.

[20] 金子学，柿倉正義，”布地のハンドリングに関する研究”，日本機械学会論文集C編，

Vol.68，No.675，pp.245-252，1998.
[21] 大澤文明，関啓明，神谷好承，”ロボットによる洗濯物の後片付け作業”，精密工学会誌，Vol.70，No.10，2004.
[22] Daisuke Sakamoto, Koichiro Honda, Masahiko Inami and Takeo Igarashi, Sketch and Run: A Stroke-based Interface for Home Robots, Proceedings of the 27th international Conference on Human Factors in Computing Systems, CHI2009, pp.197-200, Boston, USA, April4-9, 2009.
[23] Chao Sun, Bing-Kun Bao, Changsheng Xu and Tao Mei, Kinect-based visual communication system, Proceedings of the 4th International Conference on In-ternet Multimedia Computing and Service, (2012), 55.
[24] Galen Panger, Kinect in the kitchen testing depth camera interactions in practical home environments, Proceedings of the CHI'12 Extended Abstracts on Human Factors in Computing Systems , (2012), 1985.
[25] 米谷和記，三好力，”センサフュージョンを用いた個人認証”，情報処理学会第75回全国大会(2013発表予定)

学会発表

[1] 米谷和記，三好力，画像処理による靴下ペアリング手法の検討，情報処理学会第76回全国大会，2014年3月11日 - 13日.

[2] 米谷和記，三好力，複数センサを用いた靴下ペアリング手法の検討，FIT2014 第13回情報科学技術フォーラム，2014年9月3日 - 5日.

[3] Kazuki Maiya, Tsutomu MIYOSHI, Daiki Hirose, A Study of Laundry Tidiness: Socks Pairing Using Video and 3D Sensors, ICSIIT2015, 11-14 March 2015.

付録

```
#include<iostream>
#include<stdexcept>
#include<vector>
#include < vcclr.h >
#include<opencv/cv.h>
#include<opencv/highgui.h>

#include<XnCppWrapper.h>
#include"sqlite3.h"

//#include "afxinet.H"

const char* CONFIG_XML_PATH = "SamplesConfig.xml";

int flag = 1;
int center = 0;


using namespace std;
using namespace xn;
using namespace cv;
using namespace System;
using namespace System::Net;
using namespace System::Xml;
using namespace System::Drawing;
using namespace System::Windows;

//RGBピクセルの初期化
inline XnRGB24Pixel xnRGB24Pixel( int r, int g, int b){
	XnRGB24Pixel pixel = {r, g, b};
	return pixel;
}

//背景差分法関数
IplImage* BGD(IplImage* moto,IplImage* ima, int channel){

	cvSaveImage( "Data¥¥BGDmoto.png", moto);
	cvSaveImage( "Data¥¥BGDima.png", ima);

	IplImage* motoimage =
cvLoadImage("Data¥¥BGDmoto.png",CV_LOAD_IMAGE_ANYCOLOR);
	IplImage* imaimage =
cvLoadImage("Data¥¥BGDima.png",CV_LOAD_IMAGE_ANYCOLOR);
	IplImage* saimage =
cvLoadImage("Data¥¥BGDima.png",CV_LOAD_IMAGE_ANYCOLOR);
	IplImage* gray = cvCreateImage(cvSize(640, 480),
IPL_DEPTH_8U, 1);
	IplImage* niti = cvCreateImage(cvSize(640, 480),
IPL_DEPTH_8U, 1);
	IplImage* sabun = cvCreateImage(cvSize(640, 480),
IPL_DEPTH_8U, channel);


	if(channel == 3){
		int pix =
motoimage->widthStep*motoimage->height;
		for(int i=0; i<pix; i++ ) {
			int c =
imaimage->imageData[i] - motoimage->imageData[i];

	if(abs(c)<10)saimage->imageData[i] = 0;
		}
		cvCvtColor(saimage, gray, CV_BGR2GRAY);
		cvThreshold(gray, niti, 75, 255,
CV_THRESH_BINARY);
		cvErode(niti,niti,NULL,1);
		cvDilate(niti,niti,NULL,2);
		cvErode(niti,niti,NULL,1);
		cvCopy(ima,sabun,niti);

		cvSaveImage( "Data¥¥niti.png", niti);
		return sabun;
	}
	if(channel == 1){
		int pix =
motoimage->widthStep*motoimage->height;
		for(int i=0; i<pix; i++ ) {
			int c =
imaimage->imageData[i] - motoimage->imageData[i];
	if(abs(c)<2)saimage->imageData[i] = 0;
		}
		return saimage;
	}

	//cvSaveImage( "Data¥¥output1.png", saimage);
	//cvSaveImage( "Data¥¥output3.png", sabun);
}

int main(int argc, char * argv[]){
	IplImage* camera = 0;
	IplImage* depthcamera = 0;

    Mat depthshow;

	IplImage* depthc = 0;

	int flag = 0;

	try{
		//コンテキスト初期化
		Context context;
		XnStatus rc =
context.InitFromXmlFile(CONFIG_XML_PATH);
		if(rc!=XN_STATUS_OK)throw
runtime_error(xnGetStatusString(rc));

		//イメージジェネレータ作成
		ImageGenerator imagegenerator;
		rc =
context.FindExistingNode(XN_NODE_TYPE_IMAGE,imagegenerator);
		if(rc!=XN_STATUS_OK)throw
runtime_error(xnGetStatusString(rc));

		//デプスジェネレータ作成
		DepthGenerator depthgenerator;
		rc =
context.FindExistingNode(XN_NODE_TYPE_DEPTH,depthgenerator);
		if(rc!=XN_STATUS_OK)throw
runtime_error(xnGetStatusString(rc));

		//カメラサイズのイメージ作成
		XnMapOutputMode outputMode;

	imagegenerator.GetMapOutputMode(outputMode);
		camera =
cvCreateImage(cvSize(outputMode.nXRes,
outputMode.nYRes),IPL_DEPTH_8U,3);
		depthcamera =
cvCreateImage(cvSize(outputMode.nXRes,
outputMode.nYRes),IPL_DEPTH_8U,1);

		ImageMetaData imageMD;
		DepthMetaData depthMD;

		bool isDetected = true;

		//表示状態
		bool isShowImage = true;

		const int BIT = 12;

		while(1){
			//ノード更新待ち
			context.WaitAndUpdateAll();
			//画像データ取得

	imagegenerator.GetMetaData(imageMD);
			//深度データ取得

	depthgenerator.GetMetaData(depthMD);
			//デプスの座標をイメージにあ
わせる

	depthgenerator.GetAlternativeViewPointCap().SetViewPoi
nt(imagegenerator);


			//Mat
depth16(480,640,CV_16SC1,(unsigned short*)depthMD.WritableData());
			//imshow("depth16",depth16);

	//depth16.convertTo(depthshow,CV_8U,-255/4096.0,255);
			//imshow("depth",depthshow);
```

```
		//printf("%d¥n",depthshow.data[240*depthshow.step+320]);

                                                memcpy(camera->imageData,
imageMD.Data(), camera->imageSize);
                                                cvCvtColor(camera, camera,
CV_BGR2RGB);

                                                /*for(int y = 0; y < 480; ++y) {
		// ipldepthのピクセルデータへのポインタ
                                                        unsigned
char* pDepthImgData = (unsigned char*)(depthcamera->imageData + y *
depthcamera->widthStep);
		// Kinectの深度データのポインタ
                                                        short*
pKinectDepth = (short*)(depthMD.WritableData() + y *
depthcamera->widthStep);
                                                        for(int x = 0; x
< 640; ++x) {
		// Kinectの深度データを取得・スケーリング

		int depthValue = (int)(255.0 * pKinectDepth[x] / ((0x1 <<
BIT) - 1));
		// 深度データの設定（指定した範囲を超えた場合は無視）

		pDepthImgData[depthcamera->nChannels * x] =
depthValue > 255 ? (unsigned char)0 : (unsigned char)-depthValue;
                                                        }
                                                }

		printf("%d¥n",depthcamera->imageData[240*depthcamera-
>widthStep+320]);*/

// ↓ ----------------------------深度色化
---------------------------------------------------------

		memcpy(depthcamera->imageData, imageMD.Data(),
depthcamera->imageSize);
                                                for(int y = 0; y < 480; ++y) {
                                                        for(int x = 0; x
< 640; ++x) {

		int kyori = depthMD(x,y);

		int bright = (kyori)-550;

		if(bright < 0 | | bright > 255)bright = 255;

		depthcamera->imageData[y*depthcamera->widthStep+x] =
bright;
                                                        }
                                                }

		//printf("%d",depthMD(320,240));
// ↑
-----------------------------------------------------------------------------------------

// ↓ ----------------------------背景差分法適応処理
-----------------------------------------------------
                                                if(flag == 1){


		cvSaveImage("Data¥¥ima.png", camera);

		cvSaveImage("Data¥¥imadepth.png", depthcamera);
                                                        IplImage*
motoimage =
cvLoadImage("Data¥¥moto.png",CV_LOAD_IMAGE_ANYCOLOR);
                                                        IplImage*
imaimage =
cvLoadImage("Data¥¥ima.png",CV_LOAD_IMAGE_ANYCOLOR);
                                                        IplImage*
sabunimage = cvCreateImage(cvSize(640, 480), IPL_DEPTH_8U, 3);

                                                        IplImage*
motodepth =
cvLoadImage("Data¥¥motodepth.png",CV_LOAD_IMAGE_ANYCOLOR);
                                                        IplImage*
imadepth =
cvLoadImage("Data¥¥imadepth.png",CV_LOAD_IMAGE_ANYCOLOR);
                                                        IplImage*
sabundepth = cvCreateImage(cvSize(640, 480), IPL_DEPTH_8U, 1);

                                                        sabunimage =
BGD(motoimage,imaimage,3);

		cvSaveImage( "Data¥¥output4.png", sabunimage);

                                                        sabundepth =
BGD(motodepth,imadepth,1);

		cvSaveImage( "Data¥¥output5.png", sabundepth);

                                                        flag = 0;
                                                }
// ↑
-----------------------------------------------------------------------------------------



		cvShowImage("Image",camera);

		cvShowImage("Depth",depthcamera);
                                                //キーイベント
                                                char key = cvWaitKey(10);

                                                //終了する
                                                if(key == 27){ //ESCキー
                                                        break;
                                                }
                                                if(key == 13){
                                                        flag = 1;
                                                }
                                                if(key == 's'){

		cvSaveImage("Data¥¥moto.png", camera);

		cvSaveImage("Data¥¥motodepth.png", depthcamera);
                                                        center =
depthMD(320,240);

		printf("%d¥n",center);
                                                }
                                                if(key == 'e'){

		cvSaveImage("Data¥¥ima.png", camera);
                                                }
                        }
		}
		catch(exception& ex){
                        cout<<ex.what()<<endl;
		}

		cvReleaseImage(&camera);

		return 0;
}

#include<iostream>
#include<string>
#include<stdexcept>
#include<vector>
#include<opencv/cv.h>
#include<opencv/highgui.h>
#include <sstream>

using namespace cv;

int
main (int argc, char **argv){

int i;
  float *line, rho, theta;
  double a, b, x0, y0;
  IplImage *src_img_std = 0, *src_img_prob = 0, *src_img_gray = 0;
  CvMemStorage *storage;
  CvSeq *lines = 0;
  CvPoint *point, pt1, pt2;

  int sum = 0, count = 0, ave = 0;

  //画像の読み込み

  //src_img_gray = cvLoadImage ("Now¥¥Depth.png",
CV_LOAD_IMAGE_GRAYSCALE);
  //src_img_prob = cvLoadImage ("Now¥¥Depth.png",
CV_LOAD_IMAGE_COLOR);

  src_img_gray = cvLoadImage ("Now¥¥ima1.png",
CV_LOAD_IMAGE_GRAYSCALE);
  src_img_prob = cvLoadImage ("Now¥¥ima1.png",
```

```
CV_LOAD_IMAGE_COLOR);

   //src_img_gray = cvLoadImage ("Now¥¥DRGB.png",
CV_LOAD_IMAGE_GRAYSCALE);
   //src_img_prob = cvLoadImage ("Now¥¥DRGB.png",
CV_LOAD_IMAGE_COLOR);

   //高低差検出
-------------------------------------------------------------------------------
   //int pix = src_img_gray->widthStep*src_img_gray->height;
   //for(int i=0; i<pix; i++ ) {
                    // if(src_img_gray->imageData[i]!=0){
                                //    sum += src_img_gray->imageData[i];
                                //    count++;
                    // }
   //}
   //ave = (sum / count)*0.95;
   //printf("%d",ave);
   //for(int i=0; i<pix; i++ ) {
                    // if(src_img_gray->imageData[i]>ave){
                                //    src_img_gray->imageData[i] = 0;
                    // }
   //}
   //cvSaveImage("Now¥¥ima1.png", src_img_gray);
   //-------------------------------------------------------------------------------------------

   IplImage* src_img_red = cvLoadImage ("Now¥¥red.png",
CV_LOAD_IMAGE_COLOR);
   uchar p[3];
   CvPoint RedPoint[1000];
   CvPoint center, center2;
   int Pcount = 0;


   for(int y=0; y<src_img_red->height; y++ ) {
                     for(int x=0; x<src_img_red->width; x++ ) {
                                p[0] =
src_img_red->imageData[src_img_red->widthStep * y + x * 3 +0];
                                p[1] =
src_img_red->imageData[src_img_red->widthStep * y + x * 3 +1];
                                p[2] =
src_img_red->imageData[src_img_red->widthStep * y + x * 3 +2];
                                if(p[0]==0&&p[1]==0&&p[2]==255){
                                           RedPoint[Pcount].x = x;
                                           RedPoint[Pcount].y = y;
                                           Pcount++;
                                }
                     }
   }

   printf("%d,%d¥n",RedPoint[0].x,RedPoint[0].y);
   printf("%d,%d¥n",RedPoint[Pcount-1].x,RedPoint[Pcount-1].y);

   int RedX = RedPoint[Pcount-1].x - RedPoint[0].x;
   int RedY = RedPoint[Pcount-1].y - RedPoint[0].y;
   float katamuki = float(RedY)/float(RedX);
   float gyaku = -10/katamuki;

   center.x = RedPoint[Pcount-1].x - (RedX/2);
   center.y = RedPoint[Pcount-1].y - (RedY/2);

   center2.x = center.x - 10;
   center2.y = center.y - int(gyaku);

   cvCircle(src_img_red,center,1,CV_RGB (255, 255, 0),5,8,0);
   printf("%d,%d¥n", RedX/2,RedY/2);
   printf("%d,%d¥n", center.x,center.y);
   printf("%d,%d¥n", center2.x,center2.y);
   printf("%f¥n", katamuki);
   printf("%f¥n", gyaku);

   //ハフ変換のための前処理
   //cvCanny (src_img_gray, src_img_gray, 50, 200, 3);
   //cvSaveImage("Now¥¥Canny.png", src_img_gray);
   storage = cvCreateMemStorage (0);

   //確率的ハフ変換による線分の検出と検出した線分の描画
   lines = 0;
   lines = cvHoughLines2 (src_img_gray, storage,
CV_HOUGH_PROBABILISTIC, 1, CV_PI / 180, 50, 0, 10);
   printf("%d", lines->total);
   for (i = 0; i < lines->total; i++) {
      point = (CvPoint *) cvGetSeqElem (lines, i);
      cvLine (src_img_prob, point[0], point[1], CV_RGB (255, 0, 0), 3, 8, 0);
   }

   // (5)検出結果表示用のウィンドウを確保し表示する
   cvNamedWindow ("Hough_line_probalistic", CV_WINDOW_AUTOSIZE);
   cvShowImage ("Hough_line_probalistic", src_img_prob);

   cvNamedWindow ("Red", CV_WINDOW_AUTOSIZE);
   cvShowImage ("Red", src_img_red);
   cvSaveImage("Now¥¥center.png", src_img_red);

   cvSaveImage("Now¥¥ima.png", src_img_prob);
   cvWaitKey (0);

   cvDestroyWindow ("Hough_line_probalistic");
   cvReleaseImage (&src_img_prob);
   cvReleaseImage (&src_img_gray);

   cvDestroyWindow ("Red");
   cvReleaseImage (&src_img_red);

   cvReleaseMemStorage (&storage);

   return 0;
}

#include<iostream>
#include<string>
#include<stdexcept>
#include<vector>
#include<opencv/cv.h>
#include<opencv/highgui.h>
#include <sstream>

using namespace std;
using namespace cv;

//string型→char型に変換
char* StoC(string St){
                    int len = St.length();
                    char* Name = new char[len+1];
                    memcpy(Name, St.c_str(), len+1);
                    return(Name);
}

//ファイル名生成
char* SS(string ss1, string ss2, int count){
                    stringstream ss;
                    ss << ss1 << count << ss2;
                    string Result = ss.str();
                    char* Name = StoC(Result);
                    return(Name);
}

//指定フォルダ内のファイル数取得
int FindFile(char File[]){
                    HANDLE hSearch;
                    WIN32_FIND_DATA fd;
                    hSearch = FindFirstFile( File, &fd );
                    if (hSearch == INVALID_HANDLE_VALUE) return 0;
                    size_t res = 0;
                    int Fresult;
                    do {
                                        std::basic_string<TCHAR>
path(fd.cFileName);
                                        if (path != (".") && path != ("..")) {
                                                            ++res;
                                        }
                    } while (FindNextFile(hSearch, &fd));
                    FindClose( hSearch );
                    Fresult=res;
                    return(Fresult);
}

struct PhotoData{
                    char* PhotoName;
                    float Fvalue;//特徴量
};


int main(){
                    int key;
                    int Ncount = 0;
                    float Fvalue = 0;
                    float result = 0;
                    struct PhotoData PhotoData[100];
```

```
		string s0 = "Now/";
		string s1 = "Data/";
		string s2 = ".jpg";

		while(1){
				int Tcount1 = 0;
				int Tcount2 = 0;
				uchar p1,p2;
				char* NowName = SS(s0,s2,Ncount);
				char* DataName = SS(s1,s2,Ncount);


				////////////////////////////Nowフォルダ内のファイル
数取得////////////////////////////
				char Nowfolder[]="Now/*.*";

				int Fresult;

				Fresult = FindFile(Nowfolder);
				printf("ファイル数%dつ¥n",Fresult);

		////////////////////////////////////////////////////////////////////////

				////////////////////////////特徴量算出
////////////////////////////
				IplImage *img = cvLoadImage(NowName,
CV_LOAD_IMAGE_ANYDEPTH | CV_LOAD_IMAGE_ANYCOLOR);
				if( img == NULL ){
						fprintf(stderr, "no such file or
directory¥n");
						exit(-1);
				}

				IplImage *gray =
cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 1);
				IplImage *bin1 =
cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 1);
				IplImage *bin2 =
cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 1);
				cvCvtColor(img, gray, CV_BGR2GRAY);
				cvThreshold(gray, bin1, 230, 255,
CV_THRESH_BINARY);//二値化
				cvThreshold(gray, bin2, 140, 255,
CV_THRESH_BINARY);//二値化

				for(int y = 0; y < bin1->height; y++){
						for(int x = 0; x < bin1->width;
x++){
								p1 =
bin1->imageData[bin1->widthStep * y + x];

		if(p1==0)Tcount1++;
						}
				}
				for(int y = 0; y < bin2->height; y++){
						for(int x = 0; x < bin2->width;
x++){
								p2 =
bin2->imageData[bin2->widthStep * y + x];

		if(p2==0)Tcount2++;
						}
				}
				Fvalue = float(Tcount2)/float(Tcount1);
				//printf("1:%d¥n",Tcount1);
				//printf("2:%d¥n",Tcount2);
				printf("Now:%f¥n",Fvalue);
				//cvReleaseImage(&bin1);
				//cvReleaseImage(&bin2);

		////////////////////////////////////////////////////////////////////////

				////////////////////////////データの格納
////////////////////////////
				PhotoData[Ncount].PhotoName =
NowName;
				PhotoData[Ncount].Fvalue = Fvalue;

		////////////////////////////////////////////////////////////////////////

				////////////////////////////特徴量比較
////////////////////////////
				for(int i = 0; i < Ncount; i++){
						result = PhotoData[i].Fvalue -
PhotoData[Ncount].Fvalue;

		if(-0.025<result&&result<0.025){

		printf("%s:%f¥n",PhotoData[i].PhotoName,result);
						}
				}

		////////////////////////////////////////////////////////////////////////



		cvNamedWindow("Image1",CV_WINDOW_AUTOSIZE);
				cvShowImage("Image1",img);

		//cvNamedWindow("Image2",CV_WINDOW_AUTOSIZE);
				//cvShowImage("Image2",bin1);

		//cvNamedWindow("Image3",CV_WINDOW_AUTOSIZE);
				//cvShowImage("Image3",bin2);

				key = cvWaitKey(0);
				if (key == 27){
						cvDestroyWindow("Image1");
						cvReleaseImage(&img);

		//cvDestroyWindow("Image2");
						//cvReleaseImage(&bin1);

		//cvDestroyWindow("Image3");
						//cvReleaseImage(&bin2);
						break;
				}
				////////////////////////////ファイルの移動
////////////////////////////
				if(rename(NowName, DataName) == 0 ){

		PhotoData[Ncount].PhotoName = DataName;
						printf("%sを%sに移動しまし
た．¥n", NowName, DataName);
				}else{
						fprintf(stderr, "%sの移動に失
敗しました．¥n", NowName);
				}

		////////////////////////////////////////////////////////////////////////
				cvReleaseImage(&img);
				cvReleaseImage(&bin1);
				cvReleaseImage(&bin2);
				if (Fresult==1){
						cvDestroyWindow("Image1");

		//cvDestroyWindow("Image2");

		//cvDestroyWindow("Image3");
						break;
				}
				Ncount++;
		}
		return 0;
}


int main (int argc, char **argv){

  const char *imagename = "Now/0.jpg";
  Mat src_img = imread(imagename, -1);
  if(!src_img.data)
    return -1;

  const int ch_width = 260;
  const int sch = src_img.channels();
  Mat hist_img(Size(ch_width * sch, 200), CV_8UC3, Scalar::all(255));

  vector<MatND> hist(3);
  const int hist_size = 256;
  const int hdims[] = {hist_size};
  const float hranges[] = {0,256};
  const float* ranges[] = {hranges};
  double max_val = .0;

  if(sch==1) {
    calcHist(&src_img, 1, 0, Mat(), hist[0], 1, hdims, ranges, true, false);
    minMaxLoc(hist[0], 0, &max_val);
  } else {
    for(int i=0; i<sch; ++i) {
```

```
            calcHist(&src_img, 1, &i, Mat(), hist[i], 1, hdims, ranges, true, false);
            double tmp_val;
            minMaxLoc(hist[i], 0, &tmp_val);
            max_val = max_val < tmp_val ? tmp_val : max_val;
        }
    }

    Scalar color = Scalar::all(100);
    for(int i=0; i<sch; i++) {
        if(sch==3)
            color =
Scalar((0xaa<<i*8)&0x0000ff,(0xaa<<i*8)&0x00ff00,(0xaa<<i*8)&0xff0000,
0);
        hist[i].convertTo(hist[i], hist[i].type(), max_val?200./max_val:0.,0);
        for(int j=0; j<hist_size; ++j) {
            int bin_w = saturate_cast<int>((double)ch_width/hist_size);
            rectangle(hist_img,
                Point(j*bin_w+(i*ch_width), hist_img.rows),
                Point((j+1)*bin_w+(i*ch_width),
hist_img.rows-saturate_cast<int>(hist[i].at<float>(j))),
                color, -1);
        }
    }

    namedWindow("Image", CV_WINDOW_AUTOSIZE);
    namedWindow("Histogram", CV_WINDOW_AUTOSIZE);
    imshow("Image", src_img);
    imshow("Histogram", hist_img);
    waitKey(0);

    return 0;
}
```