平成２３年度　特別研究報告書

3D センサを用いた個人識別のための基礎研究

龍谷大学　理工学部　情報メディア学科

学籍番号　　T080388　　岩井　大輔

指導教員　三好　力　教授

内容梗概

ホームネットワークシステムは，住宅内の家電製品やセンサをホームサーバと呼ばれる家電を統括するシステムに接続することで，宅内外からの家電制御や省エネ等のサービスを実現するシステムである．宅内では利用者がリモコン等を操作することで家電制御を可能としているが，現状では危険な家電を特定の人に操作させない等，家電の操作を制限する場合の対応策がない．改善するためには個人識別が有効であるが個人識別システムは非常に高価であり家庭に設置するのは困難である．

そこで，本研究では，家庭内で安価に個人識別をするために，安価な 3D センサが搭載されている Kinect のセンサ情報を用いた個人識別システムを実現する．Kinect の特徴として人感センサや，対象物までの距離を測定する距離画像センサが搭載されている．人物を認識することが可能だが，人物特定は不可能である．個人識別システムを実現するために個人の特徴になるデータを決定し，決定した特徴量を用いて個人を登録，照合ができるシステムを開発し，実験により認証制度の検証を行い 3D センサにおける個人識別の有効性を確認した．

目次

1章 はじめに

1．1 研究背景

近年，情報技術が応用され，家電にネットワーク機能を取り入れた，ホームネットワークシステムと呼ばれる技術が開発・研究されている．[1]これは，テレビやエアコンなど住宅内の家電製品やセンサをホームサーバと呼ばれる家電を統括するシステムに接続することで，宅内外からの家電制御や監視，複数の家電機器の連携，省エネなどのサービスを実現するシステムである．具体的には，家庭内で家電を操作する際には，利用者がホームサーバにアクセスできるリモコンのひとつを操作することで，目的の家電の制御や複数の家電制御を実現する．また，家庭外では，携帯などインターネットが利用可能な端末を用いることで，家庭にあるホームサーバにアクセスし，帰宅前にエアコンをつけておく，外出時に鍵の施錠を確認するといった制御が可能となる．ホームネットワークシステムは既にいくつか商品化されており，今後一般家庭でも普及されると予測されている．

ホームサーバを用いる利点として，家電にある温度センサ，人感センサ，マイクセンサ等のようなセンサ情報をホームサーバで管理することで，家電や室内の状態を確認することが可能になる．欠点として，外出先では携帯端末の情報から誰が何を操作した，ということまで確認可能であるが，宅内ではリモコンを用いて家電を操作するため，誰が操作したかわからないという問題点がある．また，子供に電気ヒータを操作させない等，家電を操作する際に制限をかける必要がある場合，現状では対応策がない状態にある等がある．

一方，近年，人物の動きを認識できるテレビゲーム機のゲームデバイスKinectの家電リモコンへの応用が注目されつつある．[2][3] KinectはMicrosoftのXbox360で使える複数のセンサを搭載したゲームコントローラである．ゲームを操作するために使う入力デバイスとして，人の体の動きや音声を認識して，それをゲームのコントローラとして利用するものである．特徴として人物認識が可能であるのと，Kinectから対象物までの距離が測定できる距離画像センサが搭載されていることの2点が挙げられる．さらに，USBインタフェースがあることから，コンピュータとも接続が可能であるため，ゲームコントローラ以外の利用が多くなされている．現在はデジタルサイネージ等の実世界情報をコンピュータへと取り込み情報を付加提示できる拡張現実技術や高齢者や障害者を支援する技術が多々開発されており，今後は医療や工場，料理といった画面に触れずに操作する必要がある環境等にもKinectが活用されると予測されている．

1．2 研究目的

従来のホームネットワークシステムでは複数の家電を制御することが可能であるが，現状では危険な家電を特定の人に操作させない等，家電の操作を制限する場合の対応策がない．この問題を改善するためには個人識別が有効である．人物を特定するためには，指紋や網膜認証等の特別なセンサを用いることが一般的であるが，そのような感知センサは非常に高価であり家庭に設置するのは困難である．そこで，本研究では，家庭内で安価に個人識別をするために，安価な3Dセンサが搭載されているKinectのセンサ情報を用いた個人識別システムを実現する．個人識別システムを実現するためにコンピュータにKinectを接続した際に得られる情報を分析し個人の特徴になるデータを見つける．

1．3　論文の流れ

第 1 章では，本研究の概論について述べる．第 2 章では，情報家電と既存技術について解説する．第 3 章では，Kinect の詳細について解説する．第 4 章では，提案手法である Kinect を用いた個人識別について解説する．第 5 章では，実験目的とその方法，実験環境，実験結果を示し，それに対する考察を行う．第 6 章では，本研究のまとめと今後の課題について述べる．

2章　情報家電

2.1　情報家電とは

情報家電とはテレビやコンピュータ，DVD レコーダなどの家電機器を，ネットワークに接続したものである．家電機器をネットワーク化することで宅内外に関わらず，家電の情報やデータを取り出すことが可能になり，冷蔵庫や照明，エアコン，扇風機，DVD などの状態が常に監視，制御できるようになる．この情報家電で構成されるシステムはホームネットワークシステム（HNS）と呼ばれている．従来，ホームネットワーク環境は PC とプリンタなどの周辺機器で構成されていた．しかし，ネットワークインターフェースを備えた家電機器が多数登場してきたことで，その構成範囲を拡大してきている．

新しい構成として，複数の情報家電とホームサーバから成り立っており，ネットワーク接続がされている．ネットワークに接続された家電は，外部のソフトウェアから制御可能なインタフェースが備えられている．それに対し，ホームサーバは，情報家電を統合管理し，さらに外部ネットワークへのゲートウェイとしても働く．ホームネットワークシステムの構成を図 1 に示す．

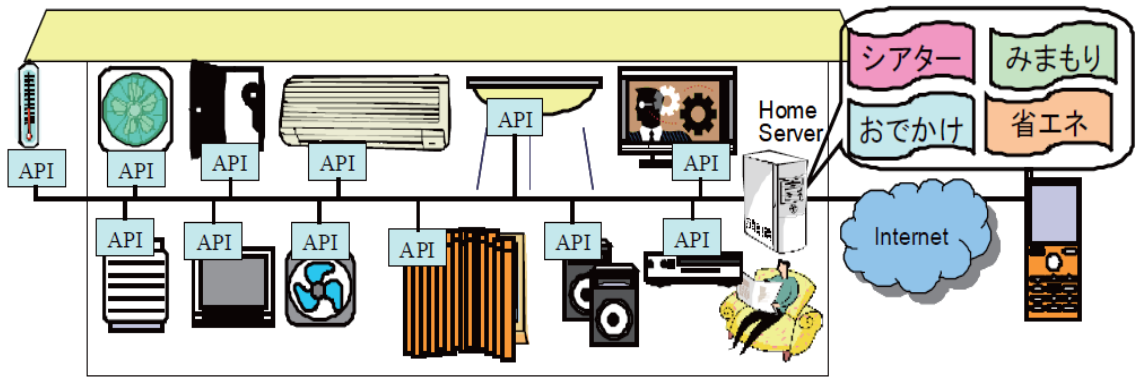

図 1：ホームネットワークシステムの構成

ホームネットワークシステムを用いた例として，ひとつの操作で自宅内を映画館のような空間を作り出し映像を視聴できるシアターモードサービスが挙げられる．これは，あらかじめホームサーバにシアターモードという名前でテレビやスピーカー，照明，カーテン設定という複数の環境設定を一度に操作できるように登録しておく．利用者がシアターモードを要求をすると，ホームサーバから家電それぞれへ制御信号が送られ，テレビが起動し，スピーカーが適切な音に調整され，照明を暗くし，カーテンが閉まる，という複数の動作を一度に自動で操作できるものである．このように本来複数の工程を必要とする作業でも，利用者への負担は最小限に抑えた操作にできる．

ホームネットワークは，各家電の通信規格や家電本来の機能のによりパソコン系，AV 家電系，白物家電系，センサ系ネットワークの 4 つの系統に分類される．まず，パソコン系ネットワークとして，現在，パソコンは家庭内のネットワーク端末としての役割を果たす装置として一翼を担っている．加えて近年，パソコンの周辺機器であるプリンタやスキャナなどにもネットワークインタフェースを備え，かつネットワークに対応するものが増えてきている．今後のネットワークサービスの主軸としての役割が期待さている，次に，AV 家電系ネットワークはデジタルテレビや DVD レコーダ，ホームサーバなどが含まれる．無線

LAN や Ethernet にも対応し，DLNA(Digital Living Network Alliance)に準拠したものもある．白物家電系はエアコンや冷蔵庫，洗濯機など，生活必需品となっている家電が含まれる，元々，各白物家電は単独でサービスを提供していたがネットワーク化によりディスプレイへの生活情報の表示や，ECHONET 規格に準拠した家電は外部から操作できるようになった．最後に，センサ系ネットワークは火災報知機や対人センサ等は本来，情報家電とは性質が異なるがネットワークを通じて、検地した情報を転送することが可能である．［4］

2．2 既存技術

2．2．1 iRemocon

既に商品化されている情報家電として iRemocon を挙げる．これは，2011 年 7 月 6 日に株式会社グラモから販売されているネットワーク型リモコン端末である．図 5 のような形状で，WiFi ルータと接続し，家電のリモコン信号を学習させることによって，iPhone・iPad から登録した家電の信号を制御することが可能になる．この学習には受信用に広帯域赤外線学習センサを用いて学習し，送信には超高出力・広角赤外線 LED を使用して家電を制御している．また，スマートフォンで家電を制御することが可能であるため，宅内だけでなく外出先からでも操作が可能となる．例えば，家に帰る前にエアコンの電源を入れておき，帰宅時に部屋を快適な温度に調整できる，見たいテレビを録画する等という操作ができる．さらに，ボタン一つで自動的にエアコンがつき，音楽が流れるサービスや，2．1 節でも述べたシアターモードサービスを提供できるマクロ機能があり，事前に登録しておくだけで複数の家電を連携させることも可能になる．他にもリモコンのデザインを自由に変更することが出来るサービスや，留守中に不審者に侵入されないように，部屋の電気を定期的に点灯させるようなタイマ機能も搭載している．［5］

図 5：iRemocon

2．2．2 フェミニティ

フェミニティは 2002 年 4 月に東芝から販売している．これは，IT ホームゲートウェイと呼ばれるホームサーバと各家電に取り付ける子機からなる．子機には業界で初めての無線技術である Bluetooth を採用し，ホームサーバと家電間の通信を実現している．利用者の必要に応じた家電をインターネット経由で制御，管理することが可能である．利用は携帯

電話でホームサーバにアクセスし家電を制御する．子機を設置できる家電にはエアコン，床暖房，照明，電力計測器，給湯機，インターホン，電気錠がある．利用例として，インターホンと電気錠、照明、エアコンに無線通信が可能となる子機を取り付けホームサーバに接続する．そうすることで外出先からも携帯を通じて鍵の施錠状態の確認と施錠ができるようになり，来訪があった際にはメールで通知することも可能な他，照明やエアコンの制御も専用のリモコンを使わずに操作が可能となる．［6］

2．2．3　wi-Go プロジェクト

wi-Go プロジェクトではショップセンター等にある買い物かごに Kinect を取り付け，車椅子や妊婦といった体が不自由な人が，かごを持たずに買い物ができるように買い物かごが対象者の後ろを自動でついていくシステムである．Kinect の第三者による開発は，コンピュータ上で遊べるゲームだけでなく，障害者や老人が特別に意識することなく使用することが出来るユニバーサルデザインにも活用されている．［7］

2．2．4　盲導犬ロボット

視覚障害者にとって盲導犬はなくてはならないパートナーであるが，現実問題として育成や維持・管理をするのは大変である．それを解決するかもしれないのが日本精工株式会社(NSK)によって開発されている盲導犬ロボットである．図 6 のような形状で，新開発の外界認識技術とガイダンス機能により、階段の位置の検出と形状認識を自律して行ない，階段を先導しながら昇降できる四脚車輪型ロボットである．このロボットは頭の部分に Kinect を使用し，内臓されている距離センサを使い階段までの距離や段差を認識している．さらに，障害物を回避する機能を搭載して 2020 年の実用化を目指している．マンション等で盲導犬を飼うのが困難である家庭で使用されるのではないかとされている．［8］

図 6：盲導犬ロボット　NR003

2．3　問題点と着眼点

2．3．1 節の iRemocon や 2．3．2 節のフェミニティにあるように，ある特定の家電製品を

ホームサーバに接続しネットワーク化することで，自宅内外に関わらず家電の使用状況を制御・管理できるシステムが商品化されている．これらは携帯端末をリモコン化して操作することでホームサーバへアクセスし家電を制御しているが，テレビやエアコン等のリモコンでは制御できない．携帯端末を利用することで宅内外どこでも家電の制御ができるという利点があるが，老人や障害者など，携帯端末を所持しない人物は家庭内で家電に付属されているリモコンを使用するのが一般的である．また，家庭内で，仮に家電に付属されているリモコンで複数家電の連携が可能になる場合，ホームシアターサービスやエアコンの制御等，利用者があるサービスを要求できる環境にあれば，利用者が誰であろうとも同じサービスしか実行できない．例えば，父，母，兄，妹の 4 人家族でホームシアターサービスを全員が要求可能である場合，父はエアコンの温度を低く設定したいが母は温度を高めに設定したい，兄は照明をつけたまま視聴したい等サービスひとつであっても利用者によってその要求は様々である．他にも，床暖房や給湯器，電気錠等の家電は操作を誤れば危険な家電も存在する．そのような家電は子供等に操作をさせないようにするのが好ましいが，特定の人物に家電を制御する場合の対応策が現状のホームネットワークシステムには実装されていない．

利用者ごとのサービスを提供したい，特定利用者の家電の操作を制御したい，という 2 点を改善するためには，あらかじめ利用者が誰であるか把握できている，即ち，個人識別をすることができれば改善できると考えられる．個人識別をするために用いられている策として，指紋認証や網膜，筆跡，声紋認証等が挙げられる．しかし，例えば指紋認証の場合，NEC 社が販売している指紋認証システムであっても，1 台約 40 万円と非常に高価であり，部屋の入退室のみの管理しか行うことができない．また，入室の際は必ず認証する必要がある上，複数人が同一の部屋に入室している場合等では役に立たない．そのような高価な感知センサを意欲的に設置しようと考える家庭は少なく，普及させるのは非常に困難である．［9］

そこで，本研究では 14800 円と安価である Kinect を用いた個人識別システムを実現する．Kinect は Xbox で使用できるゲームデバイスであるが，高機能な近赤外線距離画像センサが搭載されており，コンピュータと接続可能なことから，2．3．3 節の wi-Go プロジェクトや 2．3．4 節の盲導犬ロボットのように障害者や高齢者の負担を少なくする技術の開発が多くなされている．また，入退室という限定的な時間での利用ではなく，リアルタイムで宅内の空間を監視することが可能である．

3 章　Kinect とは

Kinect は Microsoft から販売されている Xbox360 で使える専用のゲームデバイスである．［2］2010 年 11 月 4 日に米国で販売され，同年 11 月 20 日に日本でも販売された．Kinect の特徴として，コントローラを使わずにゲームの操作ができることが挙げられる．このようなユーザインタフェースは Nattural 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が必要であるのに対し，Kinect は内臓されている 3 つのセンサ用いて高速に演算するプロセッサで処理することで，人間を検出し，プレイヤーの姿勢を認識し，リアルタイムの動きを把握しゲームの操作を可能としている．内臓されているセンサには，映像を出力したり，写真を撮るためのカラー画像を出力する映像センサカメラ，Kinect から対象物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 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赤外線を照射するレーザ照射部とレーザ照射パターンを捕らえる近赤外線カメラで構成されている．さらに，ユーザの位置を正確に捕らえるため，上下の角度を自動で調節するチルト機能も備えている．

近赤外線距離画像センサでは，Kinect から対象物までの距離を測定するために，三角測量を利用している．具体的には，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，図 2 のように，ランダムに配列したようなパターンを投射し，対象物に映ったパターンをカメラでとらえる．そして，とらえたパターンのカメラから見込み角を求める．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．［10］

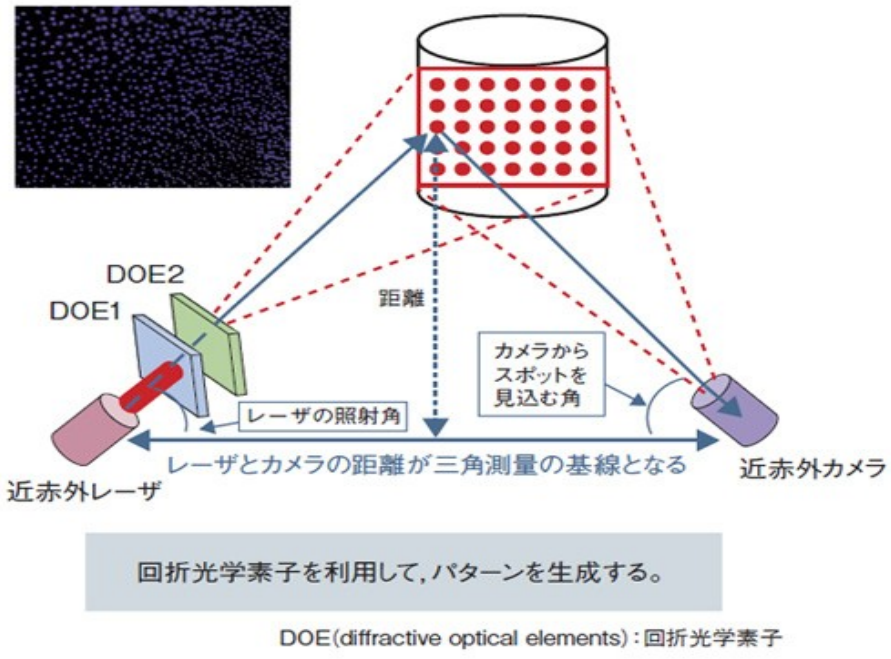

図 2：三角測量による対象物の距離測定

この距離画像センサの情報をもとに，演算処理を施し，ユーザの関節部分等の動きを追跡する．具体的な追跡箇所として，図 3 に示すように，頭，肩正面，右肩，左肩，右肘，左肘，右手首，左手首，右手，左手，骨髄，骨盤正面，右骨盤，左骨盤，右膝，左膝，右足首，左足首，右足，左足の 20 箇所の動きを検出できる．この 20 箇所をジョイントと呼び，これらの部分の動きをもとに，ユーザの全身の動きを推定している．ジョイントは同時に 2 人のユーザまで検出し，追跡することが可能である．ジョイントの各点は，Kinect の中心から縦，横方向の長さと三角測量を用いた奥行きの長さの 3 次元の値を所持している．

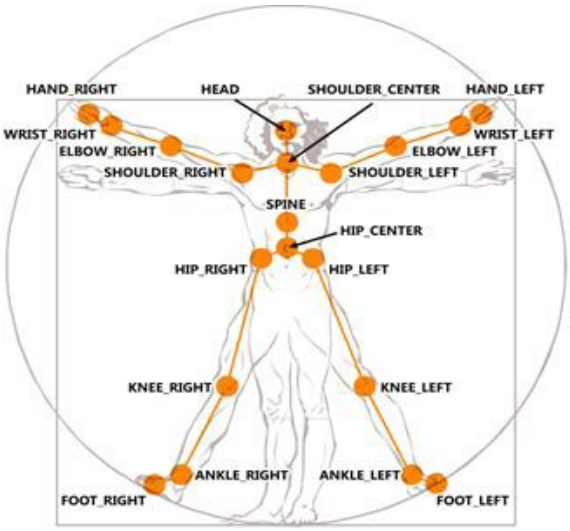

図 3：ジョイントの検出箇所

また，USB インタフェースが付属されていることから，コンピュータにも接続が可能である．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している．さらに，発売当初は Microsoft からの公式なデバイスドライバの提供予定がなかったが，2011 年 6 月 16 日に windows7 で使えるよう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．[3][11]コンピュータ上であっても Xbox360 と同じ動作が可能である．MS SDK の仕様を表 1 に示す．

表 1： MS SDK の仕様

	Kinect for Windows SDK beta
OS	Windows 7 (x86, x64)
デバイス	Kinect
ライセンス	商用利用不可
開発言語	C++, C#
RGB 解像度	1280×1024, 640×480
深度解像度	640×480

深度範囲	850mm～4000mm
ユーザトラッキングの人数	7 人
スケルトントラッキングの数	2 人
チルト制御	可能
音声	可能

MS　SDK には SDK を導入した際にすぐ実行できるサンプルが付属されている．そのサンプルのひとつである Skeletal Viewer について解説する．C++と C#の両方にサンプルがあるので C#の実行結果について解説する．Skeletal Viewer の実行結果を図 4 に示す．出力結果には 3 つの映像と数字が出力されている．まず，左上に出力されているのは，近赤外線距離画像センサより得られた Kinect から対象物までの距離を出力したものであり，対象物が近い距離にいる時を白色とし，対象物が遠くなれば遠くなるほど，黒色に近づく．また，人物が検出された時は色がつくように設計されている．対象物が取得できる深度範囲内にいない場合は白色となる．次に，右上に出力されるのが，人間を認識し，その骨格を出力するものであり，近赤外線距離画像センサの情報をから得られる 20 箇所のジョイントを出力し，人間の骨格になるようにそれぞれの関節を線でつないでいる．人間が検出されると表 1 に示したように 2 人まで関節情報を追跡し，映像をリアルタイムで更新する．左下に出力されるのは 1 秒当たりにセンサが更新された量を出力し，右下に出力されるのは映像センサカメラで出力されるカラー映像である．

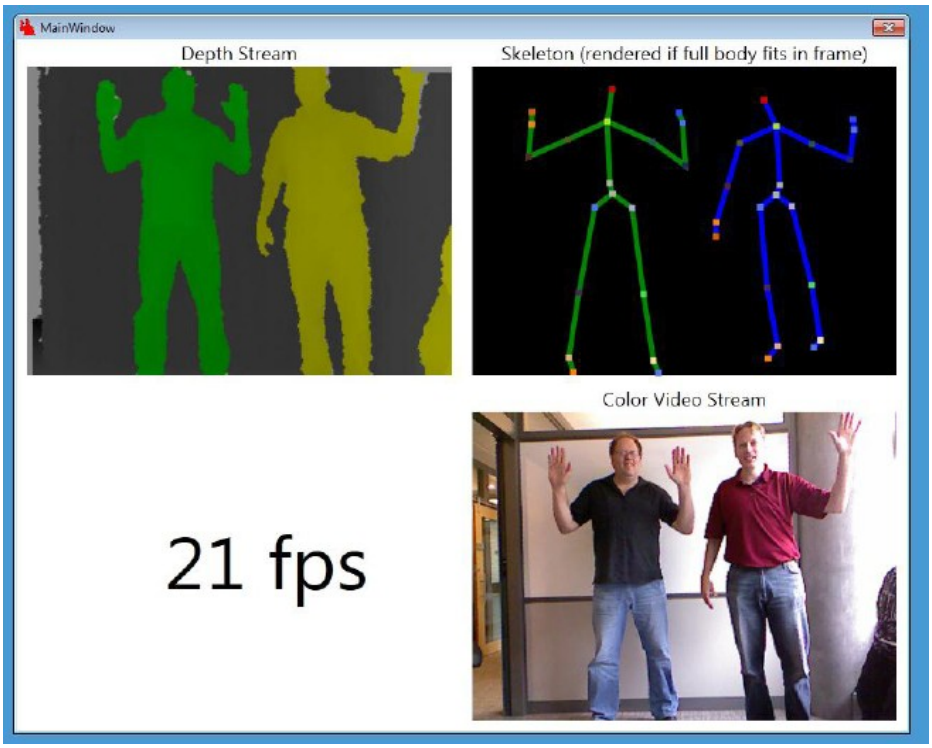

図 4：Skeletal Viewer の実行結果

4 章　3D センサを用いた個人識別

4. 1　概要

ホームシステムネットワークでは，家電同士を相互接続することで，一度に複数の家電の制御が可能であるが，宅内の場合，利用者ごとにサービスの提供したり，特定の人物に家電を制御させないためには個人識別が有効であると考えられる．本研究では，ゲームコントローラ Kinect に搭載されている近赤外線距離画像センサを用いた個人識別システムを提案する．個人識別システムを実現するために，まず，Kinect から得られるセンサ情報を用いて人間の識別に利用可能な特徴量を決定する．次に，決定した特徴量を用いた個人識別システムを開発する．

4. 2　特徴量の決定

個人識別を行うために Kinect からのどの情報が有効かを調べ、個人識別に用いる特徴量を決定する必要がある．特徴量の決定には Kinect に搭載されている近赤外線距離画像センサから得られるユーザの関節部分等の点，即ち，20 箇所のジョイントのうち 15 箇所のジョイントを利用する．ジョイントの各点は Kinect の中心から縦，横，奥行きの 3 次元の値を所持しているため，あるジョイントと別のジョイントの 2 点間の距離が算出可能である．算出結果を個人データとしてデータベースへ格納し，個人データに有効な特徴量になる長さを決定する．

開発言語は C#を用いた．開発には MS　SDK に付属されているサンプル Skeletal Viewer を元に改変した．特徴量決定の際は Skeletal Viewer の GUI 部分については，左下にある fps を削除し新たに Kinect から人間までの距離を出力するよう改変した．

プログラムについては，人間を認識し人間の骨格になるように 20 箇所のジョイントを出力しジョイントを線でつないだ，骨格情報が更新される時に動く関数を改変している．具体的には,骨格情報が更新された時に関数内で 20 箇所のジョイントの座標を記憶し，2 つのジョイントの 2 点間の距離を算出し，2，3，4m 地点に人間がいる場合，算出結果をデータベースへ書き込むという処理を追加した．骨格情報は人間が認識された際に更新頻度が約 500ms ごとに更新されるので算出結果の記録が膨大になる．このため，2，3，4m それぞれで各ジョイントのデータを集計しその平均を計算し新たにデータベースへ記録する．この計算結果をグラフで視覚化し大きく差があるものを人間の特徴量とする．

4. 3　特徴量を用いた個人識別システム

個人識別システムは 4．2 節で決定した特徴量を登録し，登録した人物を認証するシステムである．本システムは人物を登録する「登録モード」と，登録した人物を認証する「照合モード」の 2 つのモードを搭載している．

まず，登録モードでは，Kinect の近赤外線距離画像センサから人間を認識し人間の骨格になるように 20 箇所のジョイントを出力しジョイントを線でつないだ骨格情報が一人だけ追跡されている場合に限り，追跡時間が十分経過した際に，4．2 節で決定した特徴量を算出し個人データとしてデータベースへ記録する．個人データを記録する際に登録者

の名前を入力し，名前も特徴量と同様に記録する．

次に，照合モードでは登録モードで記録した特徴量データをデータベースから読み取り，近赤外線距離画像センサから骨格情報が検出された際に算出できる特徴量と，データベースに記録していた特徴量を照合する．骨格情報の検出は近赤外線距離画像センサの情報をもとに，演算処理を施して描かれるためジョイントの位置が検出される度に変わる場合がある．よってある程度誤差を含める必要がある．誤差を考慮の上，結果が等しい場合は骨格情報が検出された人物を以前にデータベースに登録された人物とし個人識別とする．

開発言語は特徴量の決定と同様に C#で行い，開発は Skeletal Viewer を改変をした．GUI 部分については左下にある fps を削除し登録と照合，通常の 3 つのボタンを取り付た．

5 章　実験と評価

本章では，　Kinect で得られる個人の特徴量を決定し，個人識別システムが妥当であるかどうか評価するための本実験を行う．

5．1　特徴量決定

5．1．1　実験概要

特徴量を決定するために，プログラムを作成し多人数から個人データを取得した．取得する個人データの内容として，ジョイントをつないだ 2 点間の距離を測定した．13 個の距離測定データを身長ごとの比でグラフ化し，人間の特徴量となるデータを決定した．人間の特徴量として，身長差がある人間を比較した場合は，足の長さや手の長さに差が出ると予測でき，肥満体型など体型に差がある人間を比較した場合は，肩幅や骨盤の長さに差が出ると予測できる．

5．1．2　実験方法

特徴量を決定するための測定データには以下の 15 箇所のジョイントを用いて，ジョイントの 2 点間の距離を合計で 13 個測定した．測定する項目を表 2 にまとめた．頭から右足首と左足首の中点までの長さである全長，頭から肩正面までの首，肩正面から骨髄までの胴，右肩から左肩までの肩幅，右肩から右肘までの右上腕，右肘から右手首までの右前腕，左肩から左肘までの左上腕，左肘から左手首までの左前腕，右骨盤から左骨盤までの腰，右骨盤から右膝までの右腿，右膝から右足首までの右向う脛，左骨盤から左膝までの左腿，左膝から左足首までの左向う脛の 13 個の距離を測定した．13 個の測定データから身長差のある人間の特徴量となるデータを決定した．

表 2：特徴量を決定する際に記録する箇所

名称	ジョイント始点	ジョイント終点
全長	頭	右足首と左足首の中点
首	頭	肩正面
胴	肩正面	骨盤正面
肩幅	右肩	左肩
右上腕	右肩	右肘
右前腕	右肘	右手首
左上腕	左肩	左肘
左前腕	左肘	左手首
腰	右骨盤	左骨盤
右腿	右骨盤	右膝

左腿	左骨盤	左膝
右向う脛	右膝	右足首
左向う脛	左膝	左足首

5. 1. 3 実験環境

特徴量を決定する際には表 3 に示したような環境で行った. Kinect から対象物までの距離は 0. 85m～4m まで測定可能であるため, 本実験では 2, 3, 4m の 1m 間隔で人物データの測定を行った. 測定距離地点が 1m では, 極端に身長が低い場合を除き, 深度情報が人間の頭から足までが捕らえることができないため, 全身が捕らえることが可能である 2m を最低の測定距離地点とした. 測定は 2 回行い骨格追跡がなされてから 2, 3, 4m の順でそれぞれ約 5 秒間静止してもらい測定を行った.

表 3:特徴量決定の実験環境

測定方法	箇所, 回数, 距離(m)
測定箇所	14 箇所
測定回数	2 回
測定距離地点	2 m, 3 m, 4 m
地面から Kinect までの高さ	0.61m

5. 1. 4 実験結果と考察

特徴量の決定には 15 人のデータを取得した. データは取得した人物の実際の身長ごとに分類した. 詳細は 1.5m までの身長が 2 人, 1.6m までの身長が 4 人, 1.7m まの身長が 6 人, 1.8m 以上の身長が 3 人である. 性別は男が 13 人, 女が 2 人である. 分類したデータの各々の平均値を求め, その結果を図 5～8 に示した. 縦軸は各測定箇所の長さを m で示したものであり, 横軸は表 2 の記録した箇所を順に全長, 首, 胴, 肩幅, 右上腕, 右前腕, 左上腕, 左前腕, 腰, 右腿, 左腿, 右向う脛, 左向う脛で 2, 3, 4m の測定箇所の結果を示している. 図 5 より, 2, 3, 4m 地点で各項目の値が全て等しくならず, 測定地点ごとに測定値がばらついていることが読み取れる. また, 図 5～8 より低身長と高身長の場合, 頭から右足首と左足首の中点である全長部分と, 右骨盤から右膝である右腿, 左骨盤から左膝である左腿の測定値に差があることがわかる. 以上の 3 つの項目を実際の身長で分類したものを図 9～11 に示した. 縦軸は測定値の長さを m で示したものであり, 横軸は実際の身長を左から右へ高くなるように並べた. 図より, 1.5m と 1.8m の人物データと比較した場合, 全長は 0.35m, 右腿は 0.19m, 左腿は 0.17m の差があることが確認できた.

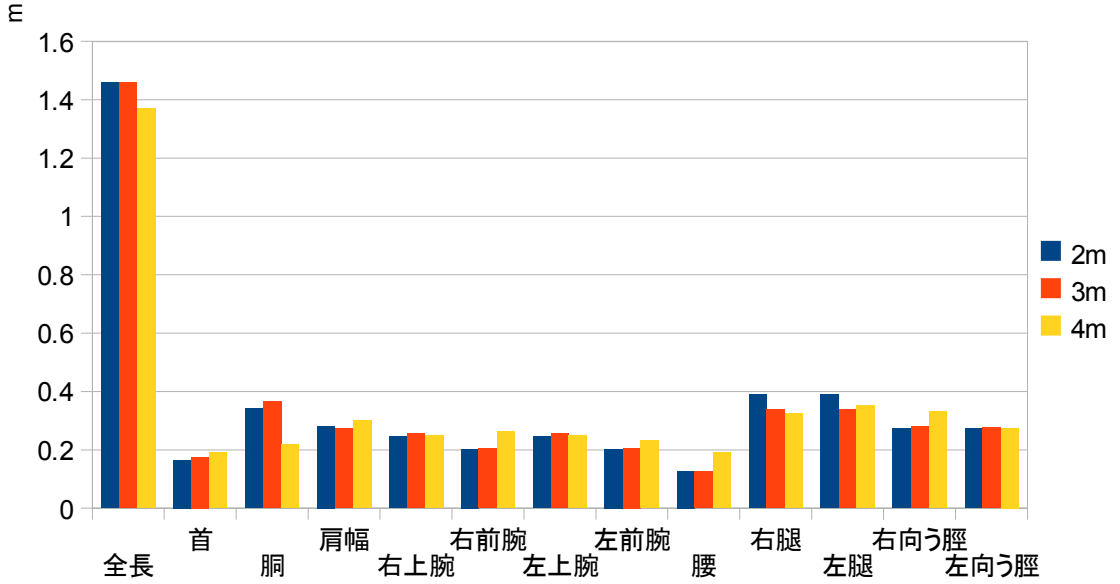

図 5:1.5m の人物データ

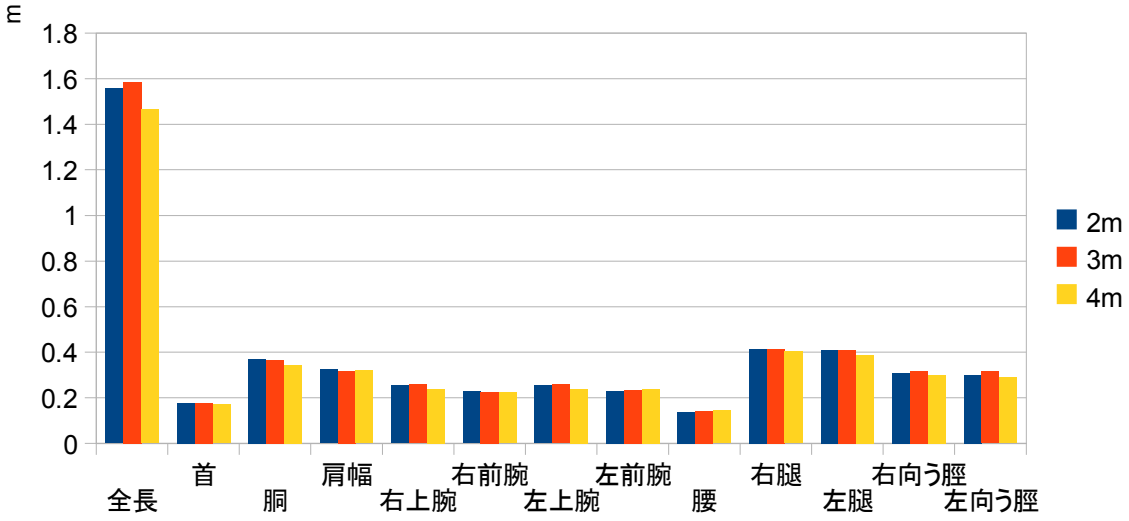

図 6:1.6m の人物データ

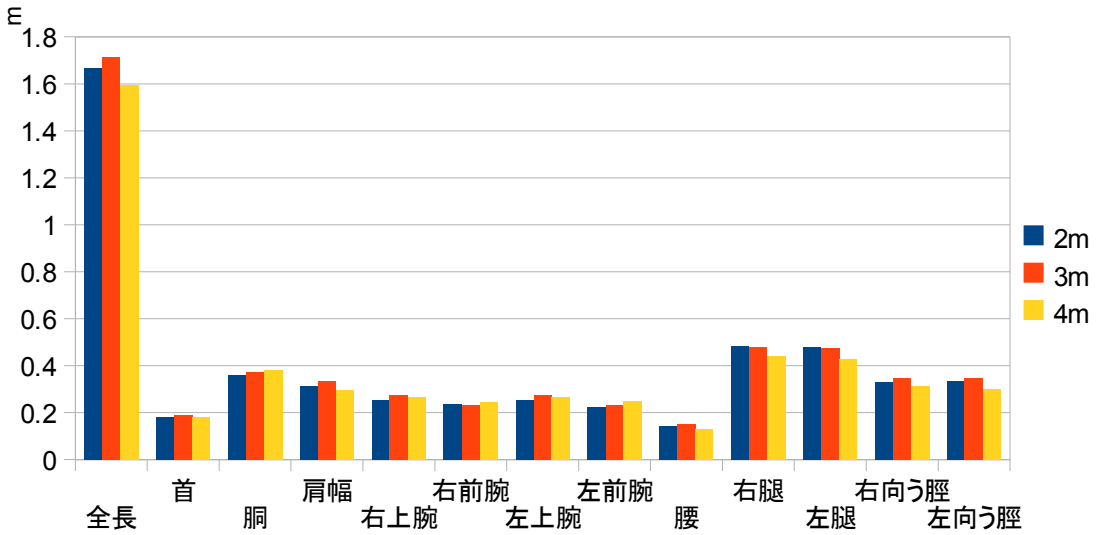

図 7:1.7m の人物データ

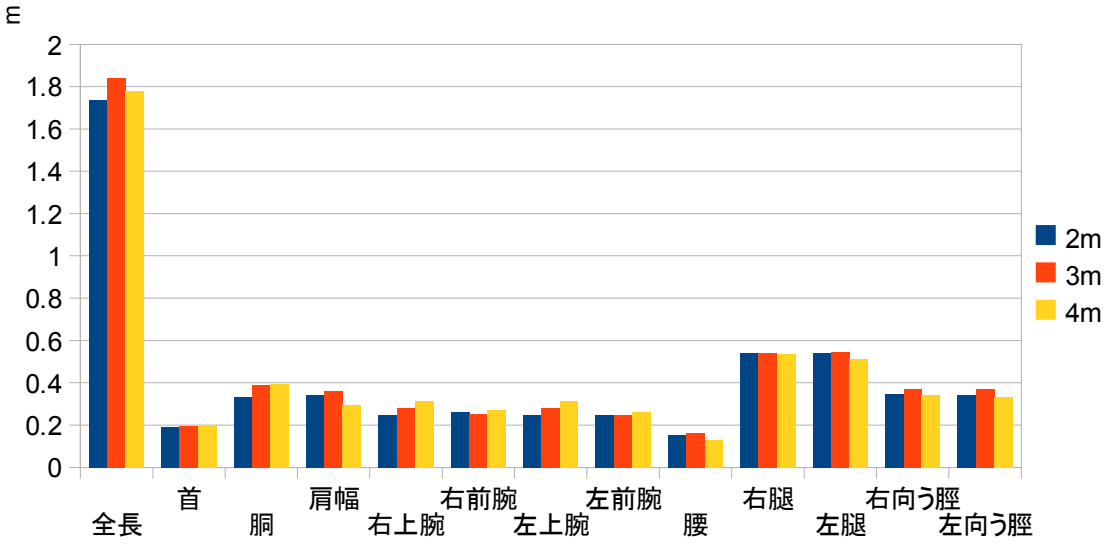

図 8:1.8m の人物データ

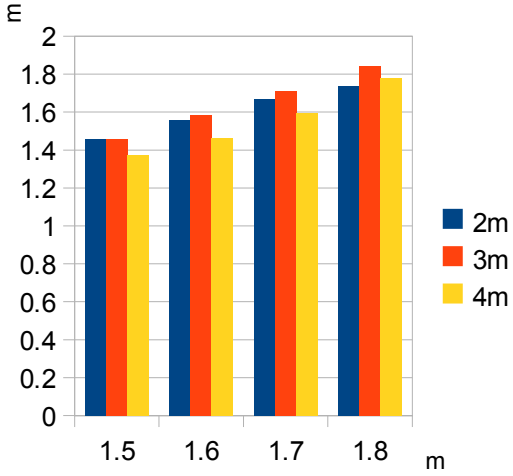

図 9:各全長の比較

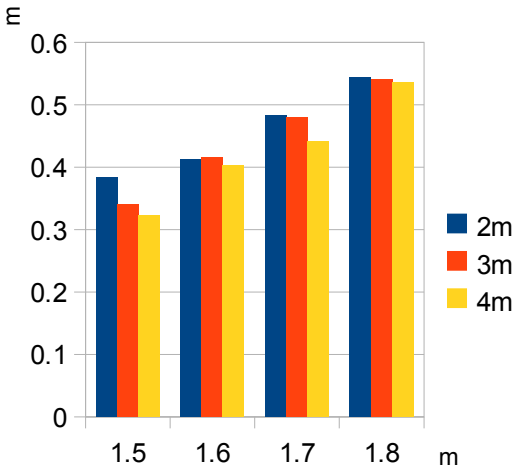

図 10:各右腿の比較

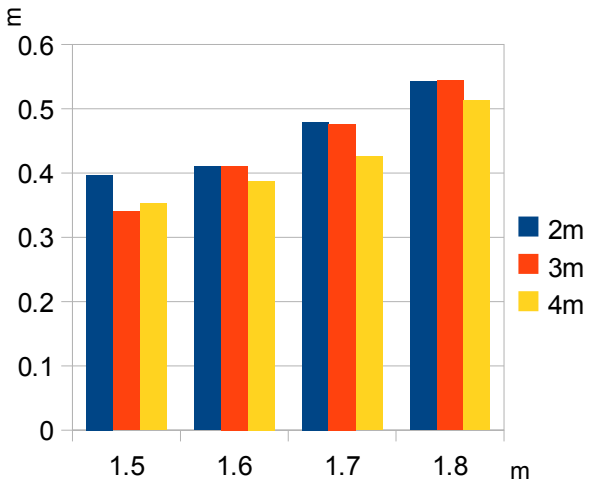

図 11:各左腿の比較

5. 2 個人識別システム

5. 2. 1 実験概要

決定した特徴量で正確に人物識別が可能かどうかを確認するために，特徴量として頭から右足首と左足首の中点である全長部分と，右骨盤から右膝である右腿，左骨盤から左膝である左腿を用いた場合の実験を行った．身長差のある人物を個人識別システムに登録しておき，照合できることを確認する．Kinect が人物を認証する度にジョイントの位置が変わる場合があるので誤差を変化させ識別率の変化量を検証した．

5. 2. 2 実験方法

ここでは，照合できることを確認するために，5．1 節で決定した特徴量を用いて，身長差がある 2 人の人物の特徴量をデータベースに記憶させ，2 人が正確に照合できることを確かめる実験を行った．

実験は次の手順で行った．Kinect が人物を認識するために，アプリケーション右下のカラー映像に 1 人で直立し身体が全て入るようにする．Kinect が人物を認識すると右上の画面にジョイントを線でつないだ骨格情報が描かれる．これで人物を認識し骨格を追跡したことが確認できる．登録はアプリケーション左下の登録ボタンをクリックし，直立のまましばらく待機しておく．するとユーザの名前を入力するフォームが出てくるので，名前を入力する．これで人物の登録は完了する．

人物の照合は照合ボタンをクリックすれば照合モードとなり，Kinect が人物を認識し骨格情報が描かれた際に描かれた人物が登録した人物かどうか調べ登録した人物であれば，登録した名前とこんにちはというメッセージを出力して照合モードを終了する．

距離画像センサから得られるジョイントの位置は同じ人物であっても人物認識される度に変わる場合があるため，特徴量に誤差を設定する必要がある．まず適当な見込み誤差を設定し個人識別を行い，どの距離で個人識別がよく成功するか検証した．次に，どの程度の見込み誤差が有効かを誤差を変化させ識別率がどのように変化するか検証した．

5. 2. 3 実験環境

個人識別システムは表 4 に示した環境で行った．人物の登録は Kinect から 3m の地点に直立し骨格情報が描かれた際に登録ボタンをクリックし，登録を行った．照合は骨格情報が描かれ，人物が 2，3，4m 地点に直立した際に照合ボタンを押し照合できるかどうか確認した．5 秒間静止しても個人を識別できない場合は人物を識別できなかったとした．測定した回数は 2，3，4m で各 10 回ずつ識別できるかどうかを調べ正確に人物が識別できるかどうかを確かめた．また，見込み誤差の初期設定は全長，右腿，左腿全て 0.1m とし，誤差の変化量は 1%刻みに変化させ測定回数は 10 回ずつ行い識別率の変化を検証した．

表 4:個人識別システムの実験環境

測定方法	回数，距離（m）
測定回数	10 回
測定距離地点	2，3，4m
地面～Kinect までの高さ	0.61m

5．2．4 実験結果と考察

どの測定距離地点で正確に個人識別がなされるかの実験結果は表 5 のようになった．表は列に測定回数を表記し，行には低身長あるいは高身長の人物が 2，3，4m の測定距離地点で個人識別ができたかどうかを示している．表の○は正しくその人物が識別された場合で，×は誤識別を表している．この表をみると，3m 地点では低身長，高身長ともに確実に識別することが可能であったが，低身長の場合 4m 地点では骨格情報が正確に表示できず，全て識別不可能だった．また，高身長の 2m 地点では身体全体がカメラに収まりきらず，骨格情報が表示されていても首から頭の部分が曲がってしまい，正確に特徴量が算出できないため識別できなかった．以上より，人物を識別するためには Kinect から 3m の地点で行うのが良いということが確認できた．

表 5:個人識別システムの実験結果

		1	2	3	4	5	6	7	8	9	10
2m	1.63m	○	○	×	○	×	○	○	×	○	○
	1.8m	×	×	×	×	×	×	×	○	×	×
3m	1.63m	○	○	○	○	○	○	○	○	○	○
	1.8m	○	○	○	○	○	○	○	○	○	○
4m	1.63m	×	×	×	×	×	×	×	×	×	×
	1.8m	○	○	×	×	○	×	○	○	×	○

次に，どの程度の見込み誤差が有効かを検証した．図 12～14 は識別システムに 1.63m の人物のみを登録した際に見込み誤差を考慮した照合率がどの程度変化したか示したものである．図 12 から順に全長，右腿，左腿の誤差と識別率・複数人を識別システムに登録した際に正確に識別したかどうかを表す正誤率の変化を表したものであり，縦軸は見込み誤差を考慮した際の照合率を％で示し，横軸は見込み誤差の値を％で示している．図より全長，右腿，左腿全ての特徴量で誤差が 1％未満の場合は識別することが不可能であり，それ以降徐々に識別できる量が増加していくことが読み取れる．1 人の人物のみを登録した場合は徐々に識別率が上昇するが，見込み誤差が増加すれば，図 12 の正誤率のように測定回数の 10 回のうち 7 回は正しく人物を照合し，残りの 3 回は目的の人物を照合するのに加えて，別の人物を誤認識してしまう場合がある．

複数人を登録した際に見込み誤差を考慮した照合率がどの程度変化したかを図 15～17 に示した．図より，全長の場合誤差が 8％を超えると 1.8m の人物と誤認識してしまい正確に人物を識別することが不可能であることが確認できた．また，右腿と左腿の場合は

6 %を超えると誤認識を起こしてしまい識別システムの意味をなさないことが確認できた.
以上により照合率が高く，誤認識がない見込み誤差は，全長は 6%以上 7%未満で右腿と左腿は 5%の見込み誤差を設定するのが望ましいと考えられる.

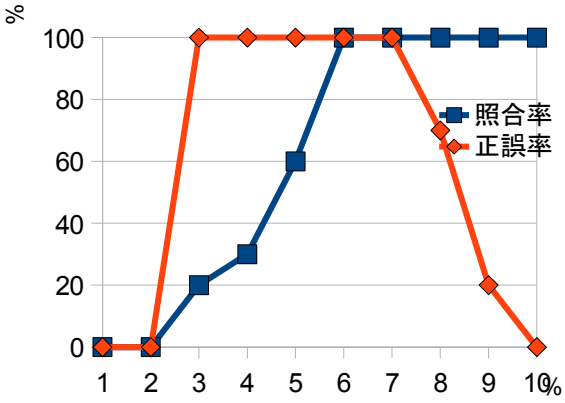

図 12:全長の照合率と正誤率

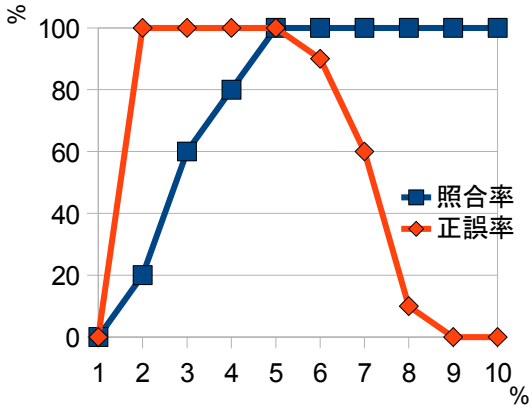

図 13:右腿の照合率と正誤率

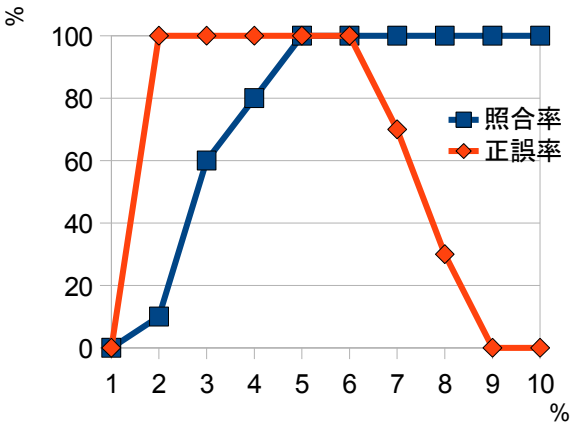

図 14 :左腿の照合率と正誤率

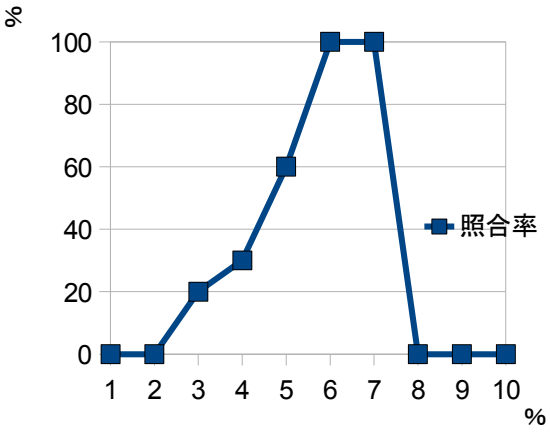

図 15:複数人の全長の照合率

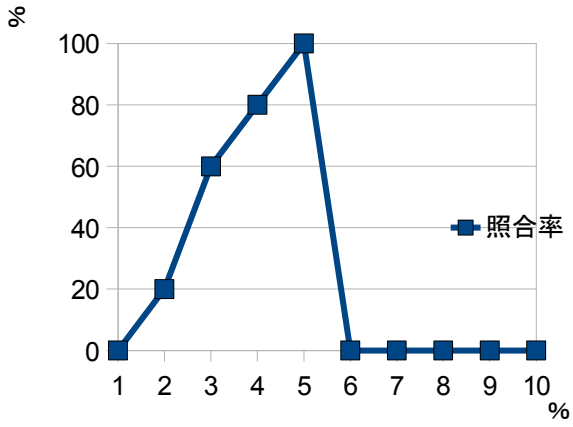

図 16:複数人の右腿の照合率

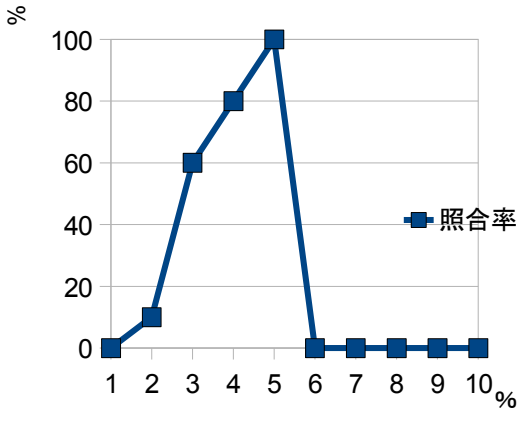

図 17:複数人の左腿の照合率

6 章　まとめ

本論文では 3D センサが搭載されている Kinect を用いた個人識別システムを提案した. 提案手法に用いる特徴量を複数のサンプルデータから, 頭から右足首と左足首の中点である全長部分と, 右骨盤から右膝である右腿, 左骨盤から左膝である左腿の 3 つと決定した. さらに, 特徴量を利用して個人識別システムが動作することを確認し, Kinect から 3m の地点で識別が最もよく識別できることが判明した. また, 個人識別の際に設定する誤差として照合率が高く, 誤認識がない見込み誤差は, 全長は 6%以上 7%未満, 右腿と左腿は 5%の見込み誤差を設定するのが望ましいと判明した.

このシステムは, ホームサーバと連携することで従来の家電ネットワークサービスの利便性を広げ, 人物ごとの家電制御を実現するのに有効であると考えられる.

今後の課題として, Kinect に搭載されているマイクセンサを用いて, 音声の高低を識別することで, 男女の識別がより明確になる可能性がある. また, Kinect は 4m までの距離でのみ骨格追跡と人間検出が可能なため, 個人識別システムを用いる空間が広い場合には利用できない. よって, 遠距離であっても人間検出や骨格追跡が可能になる必要があると考えられる. さらに, 本システムは家庭内の空間でのみ識別可能なため, 公共空間等では利用できないため, 識別範囲を広げるといった開発を進める.

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，多忙であるにも関わらず，測定データを集計に協力してくださった皆様に深く感謝いたします．

参考文献

[1] 古谷信司, 横谷哲也, 下笠清, 家電製品のネットワーク化の展開に関する一検討, 三菱電
機株式会社総合研究所（2009.3）
[2] Kinect – Xbox.com,
http://www.xbox.com/ja-JP/kinect,
[3] Microsoft Kinect SDK for Developers | Develop for the Kinect | Kinect for Windows,
http://www.microsoft.com/en-us/kinectforwindows/
[4] 小林信裕, 田辺基文, 杣信吾, 横谷哲也, 中川健治, ホームゲートウェイによる情報家電連
携サービスの提案と実現, 長岡技術科学大学（2009.05）
[5] iRemoconトップページ
http://i-remocon.com/
[6] 東芝ネットワーク家電フェミニティ,
http://feminity.toshiba.co.jp/feminity/.
[7] wi-Goプロジェクト,
http://vimeo.com/24542706,
[8] 盲導犬ロボット, http://commonpost.boo.jp/?p=18387,
[9] 指紋認証システム : 製品 | NEC,
http://www.nec.co.jp/pid/
[10] 根津禎, 日経エレクトロニクス　Kinectに見るジェスチャー入力の可能性（2010.12.27）
[11] 中村薫, Kinectセンサープログラミング（秀和システム, 2011.06.01）
[12] 古谷信司, 横谷哲也, 下笠清, 家電製品のネットワーク化の展開に関する一検討, 三菱電機株式会社総合研究所（2009.3）

付録

プログラムリスト

(1) MainWindow.xmal

```
<Window x:Class="KinectIndividualAttestationSystem.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentat
ion"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" ResizeMode="NoResize"
SizeToContent="WidthAndHeight" WindowStartupLocation="CenterScreen"
        Loaded="Window_Loaded" Closed="Window_Closed">
    <StackPanel>
        <StackPanel Orientation="Horizontal">
            <StackPanel>
                <TextBox FontSize="18" Focusable="False"
BorderThickness="0" Text="Depth Stream" HorizontalAlignment="Center"/
>
                <Image Width="400" Height="300" Name="depth"
Margin="10 0 10 10"/>
            </StackPanel>
            <StackPanel>
                <TextBox FontSize="18" Focusable="False"
BorderThickness="0" Text="Skeleton (rendered if full body fits in
frame)" HorizontalAlignment="Center"/>
                <Canvas Width="400" Height="300" Name="skeleton"
Margin="10 0 10 10" Background="Black" ClipToBounds="True"/>
            </StackPanel>

        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <StackPanel Width="400" Margin="10">
                <Button Content="登o  @録^" Height="40"
Name="button1" Width="200" FontSize="20" Margin="10"
Click="button1_Click" />
                <Button Content="照A  @合?" Height="40"
Name="button2" Width="200" FontSize="20" Margin="10"
Click="button2_Click" />
                <Button Content="通E常i" Height="40" Name="button4"
Width="200" FontSize="20" Margin="10" Click="button4_Click" />
                <TextBox Height="30" Width="350" Name="modeText"
Text="通E常iモ?ー[ドh" FontSize="18" TextAlignment="Center"/>
                <TextBox Height="60" Width="350" Name="TextBox"
Text=" " FontSize="32" TextAlignment="Center" />
            </StackPanel>
            <StackPanel>
                <TextBox FontSize="18" Focusable="False"
BorderThickness="0" Text="Color Video Stream"
HorizontalAlignment="Center" Name="VideoCaption" />
                <Image Width="400" Height="300" Name="video"
Margin="10 0 10 10"/>
            </StackPanel>
        </StackPanel>
    </StackPanel>
</Window>
```

(2)MainWindow.xmal.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Research.Kinect.Nui;
using System.Data;
using System.IO;

namespace KinectIndividualAttestationSystem
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
        Runtime nui;
        const int RED_IDX = 2;
        const int GREEN_IDX = 1;
        const int BLUE_IDX = 0;
        byte[] depthFrame32 = new byte[320 * 240 * 4];
        double spniePositionX = 0;
        double spniePositionY = 0;
        double stature = 0;
        double thighRight = 0;
        double thighLeft = 0;
        int count = 0;
        bool flagEntry = false;
        bool flagRead = false;
    int checkSkeleton = 0;
        Encoding sjisEnc = Encoding.GetEncoding("Shift_JIS");
        StreamWriter writer;
        StreamReader reader;
        double errorRangeStature = 0.08;
        double errorRangeRight = 0.05;
        double errorRangeLeft = 0.06;
        NameInput nameInput;
        ParsonCharacteristic[] parsonCharacteristic;

        Dictionary<JointID, Brush> jointColors = new
Dictionary<JointID, Brush>() {
            {JointID.HipCenter, new
SolidColorBrush(Color.FromRgb(169, 176, 155))},
            {JointID.Spine, new SolidColorBrush(Color.FromRgb(169,
176, 155))},
            {JointID.ShoulderCenter, new
SolidColorBrush(Color.FromRgb(168, 230, 29))},
            {JointID.Head, new SolidColorBrush(Color.FromRgb(200, 0,
0))},
            {JointID.ShoulderLeft, new
SolidColorBrush(Color.FromRgb(79,  84,  33))},
            {JointID.ElbowLeft, new SolidColorBrush(Color.FromRgb(84,
33,  42))},
            {JointID.WristLeft, new
SolidColorBrush(Color.FromRgb(255, 126, 0))},
            {JointID.HandLeft, new SolidColorBrush(Color.FromRgb(215,
86, 0))},
            {JointID.ShoulderRight, new
SolidColorBrush(Color.FromRgb(33,  79,  84))},
            {JointID.ElbowRight, new
SolidColorBrush(Color.FromRgb(33,  33,  84))},
            {JointID.WristRight, new
SolidColorBrush(Color.FromRgb(77,  109, 243))},
            {JointID.HandRight, new SolidColorBrush(Color.FromRgb(37,
69, 243))},
            {JointID.HipLeft, new SolidColorBrush(Color.FromRgb(77,
109, 243))},
            {JointID.KneeLeft, new SolidColorBrush(Color.FromRgb(69,
33,  84))},
            {JointID.AnkleLeft, new
SolidColorBrush(Color.FromRgb(229, 170, 122))},
            {JointID.FootLeft, new SolidColorBrush(Color.FromRgb(255,
126, 0))},
            {JointID.HipRight, new SolidColorBrush(Color.FromRgb(181,
165, 213))},
            {JointID.KneeRight, new SolidColorBrush(Color.FromRgb(71,
222,  76))},
            {JointID.AnkleRight, new
SolidColorBrush(Color.FromRgb(245, 228, 156))},
            {JointID.FootRight, new SolidColorBrush(Color.FromRgb(77,
109, 243))},
        };

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            nui = new Runtime();
            nameInput = new NameInput();
            parsonCharacteristic = new ParsonCharacteristic[100];
            for (int i = 0; i < parsonCharacteristic.Length; i++)
            {
                parsonCharacteristic[i] = new ParsonCharacteristic();
            }
            try
            {
                nui.Initialize(RuntimeOptions.UseDepthAndPlayerIndex
| RuntimeOptions.UseSkeletalTracking | RuntimeOptions.UseColor);
            }
            catch (InvalidOperationException)
            {
                System.Windows.MessageBox.Show("Runtime
initialization failed. Please make sure Kinect device is plugged
in.");
```

```
                return;
            }
            try
            {
                nui.VideoStream.Open(ImageStreamType.Video, 2,
ImageResolution.Resolution640x480, ImageType.Color);
                nui.DepthStream.Open(ImageStreamType.Depth, 2,
ImageResolution.Resolution320x240, ImageType.DepthAndPlayerIndex);
            }
            catch (InvalidOperationException)
            {
                System.Windows.MessageBox.Show("Failed to open
stream. Please make sure to specify a supported image type and
resolution.");
                return;
            }
            nui.DepthFrameReady += new
EventHandler<ImageFrameReadyEventArgs>(nui_DepthFrameReady);
            nui.SkeletonFrameReady += new
EventHandler<SkeletonFrameReadyEventArgs>(nui_SkeletonFrameReady);
            nui.VideoFrameReady += new
EventHandler<ImageFrameReadyEventArgs>(nui_VideoFrameReady);
        }
        byte[] convertDepthFrame(byte[] depthFrame16)
        {
            for (int i16 = 0, i32 = 0; i16 < depthFrame16.Length &&
i32 < depthFrame32.Length; i16 += 2, i32 += 4)
            {
                int player = depthFrame16[i16] & 0x07;
                int realDepth = (depthFrame16[i16 + 1] << 5) |
(depthFrame16[i16] >> 3);
                byte intensity = (byte)(255 - (255 * realDepth /
0x0fff));

                depthFrame32[i32 + RED_IDX] = 0;
                depthFrame32[i32 + GREEN_IDX] = 0;
                depthFrame32[i32 + BLUE_IDX] = 0;
                switch (player)
                {
                    case 0:
                        depthFrame32[i32 + RED_IDX] = (byte)
(intensity / 2);
                        depthFrame32[i32 + GREEN_IDX] = (byte)
(intensity / 2);
                        depthFrame32[i32 + BLUE_IDX] = (byte)
(intensity / 2);
                        break;
                    case 1:
                        depthFrame32[i32 + RED_IDX] = intensity;
                        break;
                    case 2:
                        depthFrame32[i32 + GREEN_IDX] = intensity;
                        break;
                    case 3:
                        depthFrame32[i32 + RED_IDX] = (byte)
(intensity / 4);
                        depthFrame32[i32 + GREEN_IDX] = (byte)
(intensity);
                        depthFrame32[i32 + BLUE_IDX] = (byte)
(intensity);
                        break;
                    case 4:
                        depthFrame32[i32 + RED_IDX] = (byte)
(intensity);
                        depthFrame32[i32 + GREEN_IDX] = (byte)
(intensity);
                        depthFrame32[i32 + BLUE_IDX] = (byte)
(intensity / 4);
                        break;
                    case 5:
                        depthFrame32[i32 + RED_IDX] = (byte)
(intensity);
                        depthFrame32[i32 + GREEN_IDX] = (byte)
(intensity / 4);
                        depthFrame32[i32 + BLUE_IDX] = (byte)
(intensity);
                        break;
                    case 6:
                        depthFrame32[i32 + RED_IDX] = (byte)
(intensity / 2);
                        depthFrame32[i32 + GREEN_IDX] = (byte)
(intensity / 2);
                        depthFrame32[i32 + BLUE_IDX] = (byte)
(intensity);
                        break;
                    case 7:
                        depthFrame32[i32 + RED_IDX] = (byte)(255 -
intensity);
                        depthFrame32[i32 + GREEN_IDX] = (byte)(255 -
intensity);
                        depthFrame32[i32 + BLUE_IDX] = (byte)(255 -
intensity);
                        break;
                }
            }
            return depthFrame32;
        }

        void nui_DepthFrameReady(object sender,
ImageFrameReadyEventArgs e)
        {
            PlanarImage Image = e.ImageFrame.Image;
            byte[] convertedDepthFrame =
convertDepthFrame(Image.Bits);

            depth.Source = BitmapSource.Create(
                Image.Width, Image.Height, 96, 96,
PixelFormats.Bgr32, null, convertedDepthFrame, Image.Width * 4);
        }

        private Point getDisplayPosition(Joint joint)
        {
            float depthX, depthY;
            nui.SkeletonEngine.SkeletonToDepthImage(joint.Position,
out depthX, out depthY);
            depthX = Math.Max(0, Math.Min(depthX * 320, 320));
//convert to 320, 240 space
            depthY = Math.Max(0, Math.Min(depthY * 240, 240));
//convert to 320, 240 space
            int colorX, colorY;
            ImageViewArea iv = new ImageViewArea();
            nui.NuiCamera.GetColorPixelCoordinatesFromDepthPixel(
                ImageResolution.Resolution640x480, iv, (int)depthX,
(int)depthY, (short)0, out colorX, out colorY);
            return new Point((int)(skeleton.Width * colorX / 640.0),
(int)(skeleton.Height * colorY / 480));
        }

        Polyline
getBodySegment(Microsoft.Research.Kinect.Nui.JointsCollection joints,
Brush brush, params JointID[] ids)
        {
            PointCollection points = new PointCollection(ids.Length);
            for (int i = 0; i < ids.Length; ++i)
            {
                points.Add(getDisplayPosition(joints[ids[i]]));
            }
            Polyline polyline = new Polyline();
            polyline.Points = points;
            polyline.Stroke = brush;
            polyline.StrokeThickness = 5;
            return polyline;
        }

        void nui_SkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)
        {
            SkeletonFrame skeletonFrame = e.SkeletonFrame;
            int iSkeleton = 0;
            checkSkeleton = 0;
            Brush[] brushes = new Brush[6];
            brushes[0] = new SolidColorBrush(Color.FromRgb(255, 0,
0));
            brushes[1] = new SolidColorBrush(Color.FromRgb(0, 255,
0));
            brushes[2] = new SolidColorBrush(Color.FromRgb(64, 255,
255));
            brushes[3] = new SolidColorBrush(Color.FromRgb(255, 255,
64));
            brushes[4] = new SolidColorBrush(Color.FromRgb(255, 64,
255));
            brushes[5] = new SolidColorBrush(Color.FromRgb(128, 128,
255));
            skeleton.Children.Clear();
            foreach (SkeletonData data in skeletonFrame.Skeletons)
            {
                if (SkeletonTrackingState.Tracked ==
data.TrackingState)
                {
                    Brush brush = brushes[iSkeleton %
brushes.Length];
                    skeleton.Children.Add(getBodySegment(data.Joints,
brush, JointID.HipCenter, JointID.Spine, JointID.ShoulderCenter,
JointID.Head));
```

```
                    skeleton.Children.Add(getBodySegment(data.Joints,
brush, JointID.ShoulderCenter, JointID.ShoulderLeft,
JointID.ElbowLeft, JointID.WristLeft, JointID.HandLeft));
                    skeleton.Children.Add(getBodySegment(data.Joints,
brush, JointID.ShoulderCenter, JointID.ShoulderRight,
JointID.ElbowRight, JointID.WristRight, JointID.HandRight));
                    skeleton.Children.Add(getBodySegment(data.Joints,
brush, JointID.HipCenter, JointID.HipLeft, JointID.KneeLeft,
JointID.AnkleLeft, JointID.FootLeft));
                    skeleton.Children.Add(getBodySegment(data.Joints,
brush, JointID.HipCenter, JointID.HipRight, JointID.KneeRight,
JointID.AnkleRight, JointID.FootRight));
                    foreach (Joint joint in data.Joints)
                    {
                        Point jointPos = getDisplayPosition(joint);
                        Line jointLine = new Line();
                        jointLine.X1 = jointPos.X - 3;
                        jointLine.X2 = jointLine.X1 + 6;
                        jointLine.Y1 = jointLine.Y2 = jointPos.Y;
                        jointLine.Stroke = jointColors[joint.ID];
                        jointLine.StrokeThickness = 6;
                        skeleton.Children.Add(jointLine);
                    }
                    double HeadX =
data.Joints[JointID.Head].Position.X;
                    double HeadY =
data.Joints[JointID.Head].Position.Y;
                    double HeadZ =
data.Joints[JointID.Head].Position.Z;
                    double SpineX =
data.Joints[JointID.Spine].Position.X;
                    double SpineY =
data.Joints[JointID.Spine].Position.Y;
                    double SpineZ =
data.Joints[JointID.Spine].Position.Z;
                    double ShoulderCenterX =
data.Joints[JointID.ShoulderCenter].Position.X;
                    double ShoulderCenterY =
data.Joints[JointID.ShoulderCenter].Position.Y;
                    double ShoulderCenterZ =
data.Joints[JointID.ShoulderCenter].Position.Z;
                    double HipCenterX =
data.Joints[JointID.HipCenter].Position.X;
                    double HipCenterY =
data.Joints[JointID.HipCenter].Position.Y;
                    double HipCenterZ =
data.Joints[JointID.HipCenter].Position.Z;
                    double HipRightX =
data.Joints[JointID.HipRight].Position.X;
                    double HipRightY =
data.Joints[JointID.HipRight].Position.Y;
                    double HipRightZ =
data.Joints[JointID.HipRight].Position.Z;
                    double HipLeftX =
data.Joints[JointID.HipLeft].Position.X;
                    double HipLeftY =
data.Joints[JointID.HipLeft].Position.Y;
                    double HipLeftZ =
data.Joints[JointID.HipLeft].Position.Z;
                    double KneeRightX =
data.Joints[JointID.KneeRight].Position.X;
                    double KneeRightY =
data.Joints[JointID.KneeRight].Position.Y;
                    double KneeRightZ =
data.Joints[JointID.KneeRight].Position.Z;
                double KneeLeftX =
data.Joints[JointID.KneeLeft].Position.X;
                    double KneeLeftY =
data.Joints[JointID.KneeLeft].Position.Y;
                    double KneeLeftZ =
data.Joints[JointID.KneeLeft].Position.Z;
                    double AnkleRightX =
data.Joints[JointID.AnkleRight].Position.X;
                    double AnkleRightY =
data.Joints[JointID.AnkleRight].Position.Y;
                    double AnkleRightZ =
data.Joints[JointID.AnkleRight].Position.Z;
                    double AnkleLeftX =
data.Joints[JointID.AnkleLeft].Position.X;
                    double AnkleLeftY =
data.Joints[JointID.AnkleLeft].Position.Y;
                    double AnkleLeftZ =
data.Joints[JointID.AnkleLeft].Position.Z;
                    stature = distance(HeadX, HeadY, HeadZ,
ShoulderCenterX, ShoulderCenterY, ShoulderCenterZ) +
                        distance(HeadX, HeadY, HeadZ, ((AnkleLeftX +
AnkleRightX) / 2), ((AnkleLeftY + AnkleRightY) / 2), ((AnkleLeftZ +
AnkleRightZ) / 2));
                    thighRight =
distance(HipRightX,HipRightY,HipRightZ,KneeRightX,KneeRightY,KneeRigh
tZ);
                    thighLeft = distance(HipLeftX, HipLeftY,
HipLeftZ, KneeLeftX, KneeLeftY, KneeLeftZ);
                    if (flagEntry == true && data.UserIndex == 254)
                    {
                        count++;
                        if (count == 100)
                        {
                            nameInput.Show();
                            nameInput.Closed += new
EventHandler(nameInput_Closed);
                            count = 0;
                            flagEntry = false;
                            modeText.Text = "通常モード";
                        }
                    }
                    if (flagRead == true && data.UserIndex == 254)
                    {
                        int i = 0;
                        while (parsonCharacteristic[i].Name != "-1")
                        {
                            if ((parsonCharacteristic[i].Stature -
(parsonCharacteristic[i].Stature * errorRangeStature) < stature &&
stature < parsonCharacteristic[i].Stature +
(parsonCharacteristic[i].Stature * errorRangeStature)) &&
                                (parsonCharacteristic[i].ThighRight -
(parsonCharacteristic[i].ThighRight * errorRangeRight) < thighRight
&& thighRight < parsonCharacteristic[i].ThighRight +
(parsonCharacteristic[i].ThighRight * errorRangeRight)) &&
                                (parsonCharacteristic[i].ThighLeft -
(parsonCharacteristic[i].ThighLeft * errorRangeLeft) < thighLeft &&
thighLeft < parsonCharacteristic[i].ThighLeft +
(parsonCharacteristic[i].ThighLeft * errorRangeLeft))
                                )
                            {
                                MessageBox.Show("識別完了¥n" +
parsonCharacteristic[i].Name + "さん　こんにちは");
                                flagRead = false;
                                modeText.Text = "通常モード";
                                reader.Close();
                                break;
                            }
                            i++;
                        }
                    }
                }
                iSkeleton++;
            }
        }

        void nameInput_Closed(object sender, EventArgs e)
        {
            if (Name != null)
            {
                writer.WriteLine(Name);
                writer.WriteLine(stature);
                writer.WriteLine(thighRight);
                writer.WriteLine(thighLeft);
                writer.Close();
            }
            nameInput = new NameInput();
        }

        double distance(double x1, double y1, double z1, double x2,
double y2, double z2)
        {
            double x, y, z;
            x = x1 - x2;
            y = y1 - y2;
            z = z1 - z2;
            return Math.Sqrt(x * x + y * y + z * z);
        }

        void nui_VideoFrameReady(object sender,
ImageFrameReadyEventArgs e)
        {
            PlanarImage Image = e.ImageFrame.Image;
            video.Source = BitmapSource.Create(
                Image.Width, Image.Height, 96, 96,
PixelFormats.Bgr32, null, Image.Bits, Image.Width *
Image.BytesPerPixel);
            checkSkeleton++;
            if (checkSkeleton == 10)
            {
```

```
                checkSkeleton = 0;
                TextBox.Text = ".";
                skeleton.Children.Clear();
            }
        }

        private void Window_Closed(object sender, EventArgs e)
        {
            nui.Uninitialize();
            Environment.Exit(0);
            if (flagEntry == true)
            {
                writer.Close();
            }
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            if (flagRead == true)
            {
                reader.Close();
            }
            if (flagEntry == true)
            {
                writer.Close();
            }
            modeText.Text = "登録モードh";
            flagEntry = true;
            flagRead = false;
            writer = new StreamWriter(@"C:¥Visual Studio
2010¥KinectMemory.txt", true, sjisEnc);
        }

        private void button2_Click(object sender, RoutedEventArgs e)
        {
            modeText.Text = "照合モード";
            if (flagEntry == true)
            {
                writer.Close();
            }
            if (flagRead == true)
            {
                reader.Close();
            }
            flagEntry = false;
            flagRead = true;
            reader = new StreamReader(@"C:¥Visual Studio
2010¥KinectMemory.txt", sjisEnc);
            int i = 0;
            int j = 0;
            String tmp;
            while (reader.Peek() >= 0)
            {
                tmp = reader.ReadLine();
                switch (i % 4)
                {
                    case 0:
                        parsonCharacteristic[j].Name = tmp;
                        break;
                    case 1:
                        parsonCharacteristic[j].Stature =
double.Parse(tmp);
                        break;
                    case 2:
                        parsonCharacteristic[j].ThighRight =
double.Parse(tmp);
                        break;
                    case 3:
                        parsonCharacteristic[j].ThighLeft =
double.Parse(tmp);
                        j++;
                        break;
                }
                i++;
            }
            parsonCharacteristic[j].Name = "-1";
        }

        private void button4_Click(object sender, RoutedEventArgs e)
        {
            modeText.Text = "通常モード";
           flagEntry = false;
            flagRead = false;
        }

        public static String Name { set; get; }
    }
}
```

(3) NameInput.xaml

```
<Window x:Class="KinectIndividualAttestationSystem.NameInput"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentat
ion"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="NameInput" Height="160" Width="305">
    <StackPanel>
        <TextBlock Height="23" Name="textBlock1" FontSize="16" Text="
名前を入力してください" TextAlignment="Center" Margin=" 0 5 0 5"/>
        <TextBox Height="35" Name="textBox1" Width="150" Margin="0 5
0 5" FontSize="24"/>
        <StackPanel Orientation="Horizontal">
            <Button Content="OK" Height="30" Name="button1"
Width="100" Margin="20 0 20 0" Click="button1_Click"/>
            <Button Content="CANCEL" Height="30" Name="button2"
Width="100" Margin="20 0 20 0 " Click="button2_Click"/>
        </StackPanel>
    </StackPanel>
</Window>
```

(4) NameInput.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace KinectIndividualAttestationSystem
{
    public partial class NameInput : Window
    {
        public NameInput()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            MainWindow.Name = textBox1.Text;
            MessageBox.Show(textBox1.Text + "さんを登録しました");
            this.Close();
        }

        private void button2_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
    }
}
```

(5) ParsonCharacteristic.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace KinectIndividualAttestationSystem
{
    class ParsonCharacteristic
    {
        private String name;
        private double stature;
        private double thighRight;
        private double thighLeft;

        public String Name { get; set; }
        public double Stature { get; set; }
        public double ThighRight { get; set; }
        public double ThighLeft { get; set; }
    }
}
```