「Kinect を用いた立体物の計測と PCL による 3 次元点群処理」

2013 年 2 月 5 日

京都産業大学コンピュータ理工学部ネットワークメディア学科
西浦　直樹

要約

本論文では、立体物の形状を入力し、取得した 3 次元点群データを処理する技術について研究した。研究には、Xbox360 向けにマイクロソフトが製品化した Kinect と Kinect SDK、そしてオープンソースソフトウェアの PCL(Point Cloud Library)を用いた。まず、3 次元点群データを取得するために Kinect が出力する奥行き情報を 3 次元座標に変換した。このとき、RGB カメラが取得するカラー画像と 3 次元座標を整列させる。取得した複数の 3 次元点群データを、PCL を用いて、ひとつに統合する。このとき、点群データのはずれ値を除去し、サンプリング数を削減した。そして複数視点からの点群データを SAC-IA(Sample Consensus Initial Alignment)によって初期位置合わせした後、ICP(Interative Closest Point)アルゴリズムによって高精度に位置合わせをした。

目次

1 章　序論

1.1　背景と目的

ジャイロセンサが搭載されたNintendo 3DSや、タッチ入力を行うスマートフォンなど、直感的に操作するデバイスが多く存在する。Kinectは Xbox 360と接続して使用するそのようなヒューマンインタフェースデバイスのひとつである。

Kinectは深度センサとRGBカメラを備えている。カラー画像を取得するだけでなく、立体物の3次元形状を獲得することができる。従来、3次元形状を測定する装置は高価であったが、Kinect を利用することができれば、格段にコストダウンすることが可能である。

従来の3次元形状測定装置を利用する場合、測定された生データ（3次元スキャンデータ）を処理する専用のソフトウエアが必要である。このソフトウエアは、3次元スキャンデータからパラメトリックソリッドモデルを生成するための専用ソフトウエアである。価格が高価で操作も高度である。

一方、Kinect のような安価で高速な 3 次元入力装置の出現は、これを移動ロボットなどの用途に利用する研究分野を活性化させている。その成果のひとつとして、Kinect が出力する生データを処理するオープンソースソフトウエア PCL（Point Cloud Library）が開発された。PCLは、上記の専用ソフトウエアの機能を要素別にライブラリ化したようなものである。したがって、フリーであるだけでなく、自由にアプリケーションに組み込んで利用するこが可能である。つまり、Kinect と PCLを用いれば少ない費用で3次元形状を入力、モデリングすることが可能となる。

少ない費用で立体物のモデリングができることで、実物から3Dモデルを作成することの敷居が下がり、3D モデルの利用も容易になる。例えば、ネットでの販売において商品のプレビューに実物の3Dモデルを表示することができれば、消費者は正確な形状を理解することができる。

本論文では、実物の3Dモデルの作成を容易に行うために、Kinectを用いて立体物の3次元形状を入力し、取得した3次元データを PCLで処理する技術について研究した。

1.2　論文の構成

2章は従来の立体形状の入力技術、それを踏まえた本研究のたち位置につ

いて書く。3 章は Kinect を用いた立体形状の取得に関する技術を述べる。4 章はＰＣＬを用いた 3 次元点群データ処理を、フィルタリングと位置合わせを中心に述べる。5 章は本研究の結論を書く。

2 章　立体形状の入力技術

2.1　3 次元スキャン技術

通常、機械装置の立体図面は３次元ＣＡＤシステムなどを用いて人手で作成する。このため、多くの時間と労力が必要である。しかし、3 次元スキャナを用いれば、立体物から３次元形状を取得することができ、労力を減らすことができると考えられる。また、人の立ち入ることができない場所でも 3 次元スキャナがあれば測定することができる。

3 次元スキャナには高精度であるがスキャンできる大きさが限られている接触型と、精度は落ちるが読み取り速度の早い非接触型（光学式）のものがある。現在は非接触型が主流となっている。非接触型の３次元スキャニングの測定原理には、主に三角測量を用いる。

図１は三角測量の原理を表している。赤外光プロジェクタからレーザ光やパターン光を立体物に照射する。照射したレーザ光やパターン光が立体物で反射したものを赤外光カメラで捕らえる。このとき、プロジェクタとカメラ間の距離を三角測量の基線とする。基線と、プロジェクタの照射角と、カメラの角度がわかればセンサから立体物の奥行きを求めることができる。

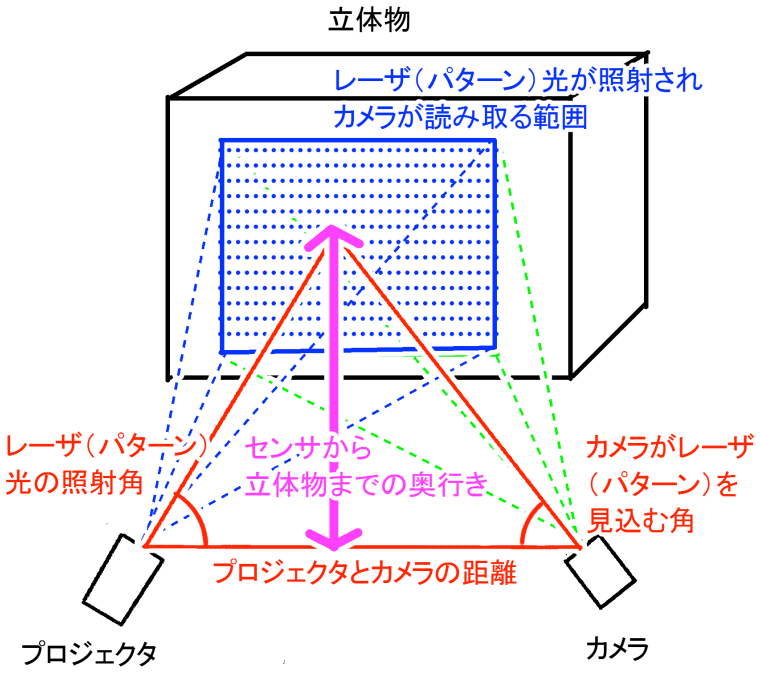

図 1　非接触型 3 次元測定装置における三角測量の原理

レーザを用いた三角測量法に基づく装置は複雑な機構であるため、3 次元スキャナは高価な機材であった。そこで Kinect の登場である。

DERiVE[1]では、Kinec センサアクティブステレオ技術が、広範囲撮影が可能でかつ動画で撮影でき、商用で使うにも 2 万円前後で買えるものとして登場したというのは革命だった、と述べている。

また、谷尻は文献[2]で、「用途を限定すれば Kinect も 3 次元形状の計測装置として十分に使えるのではないか」と述べている。

Kinect の安価で高速な 3 次元スキャナとしての利用が期待されている。

2.2　3 次元点群データの処理技術

3 次元スキャナーによって取得した、3 次元形状を表す生データ（3 次元点群データ）には不必要なものが含まれている場合がある。そのようなデータに対してノイズを除去するフィルタリングを行わなければならない。また、取得した複数の 3 次元点群データの位置合わせや、メッシュを張ることも 3 次元点群データの利用には必要になる場合がある。このような 3 次元点群データを処理する場合は、専用のソフトウェアを使用する。

3 次元点群データの処理に使うソフトウェアも高価なものであったが、近年 RapidformXOM、PCL といったフリーで使用できるソフトウェアやライブラリがリリースされた。これらは安価で高速に 3 次元点群データを取得することのできる Kinect の出現により注目が高まっている。

図 2 は PCL が備えている 3 次元点群データ処理を行うためのモジュール一覧である。このように、フィルタ、3 次元特徴量、3 次元キーポイント、kd-tree や octree といった探索のための構造、探索アルゴリズム、レジストレーション、RANSAC などによるフィッティング、セグメンテーション、サーフェス処理、データの可視化、距離画像の生成の機能が提供されている。

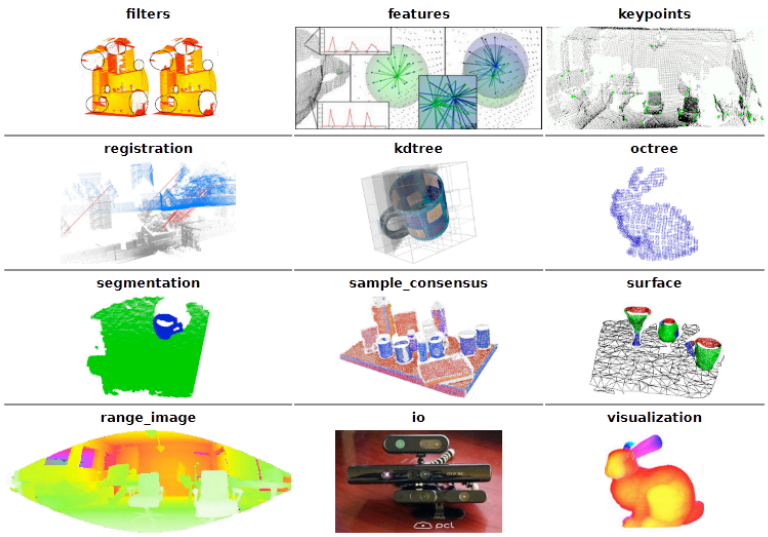

図 2　様々な点群データ処理

2.3　KinectFusion

Kinect で 3 次元形状の入力を行うソフトウェアのひとつに、現在開発が行われている KinectFusion がある。ストリーミング入力からリアルタイムで周囲の環境をモデリングできることが特徴である。

KinectFusion の仕組みは次のようになっている。Kinect のセンサから得たデプス画像とカラー画像を即座に 3 次元点群データに変換し、保持する。□フレーム前のデータと取得したばかりのデータを比べて ICP（Interative Closest Point）アルゴリズムによる位置合わせを行う。データ数が膨大に膨れることを防ぐためにボクセルグリッドフィルタによってデータ数を低減する。

KinectFusion は周囲の環境を正確にモデリングできるため、物理シュミレーションに応用することも可能となっている。3D スキャナとして静止したオブジェクトを取り込むために Kinect を用いる技術はあったが、リアルタイムで 3 次元形状の入力を次々に行うことができるソフトウェアはなかった。また KinectFusion は環境が変化した際、即座に新しいデータの更新を行うことができる。

図 3 は KinectFusion の動作画面である。左上では入力された法線ベクトル、右上では入力されたカラー画像、左下では面ごとに統合した法線ベクトル、右下では 3 次元形状をそれぞれ出力していることがわかる。

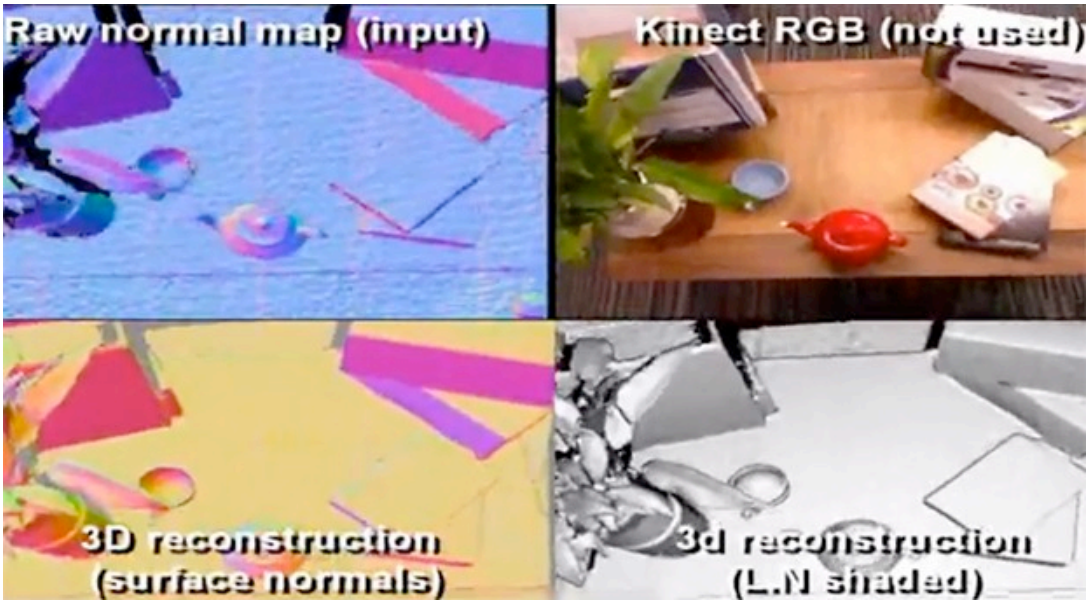

図 3　KinectFusion の動作画面

2.4　本研究の位置づけ

本研究では、KinectFusion のように Kinect を 3 次元点群データの入力装置として活用し、PCL を用いて 3 次元形状をモデリングを行うシステムを開発する。

3 次元スキャナで立体形状の生データを入力してから、メカニカル CAD で加工が可能な完全な 3D モデルを生成するまでのプロセスは、図 4 のようになる。本研究ではこの図 4 のうち、複数データの位置合わせまでを開発した。開発には Kinect と PCL のほかにマイクロソフトから提供されている Kinect SDK、そして VisualStudio 2010 の C++を使用する。

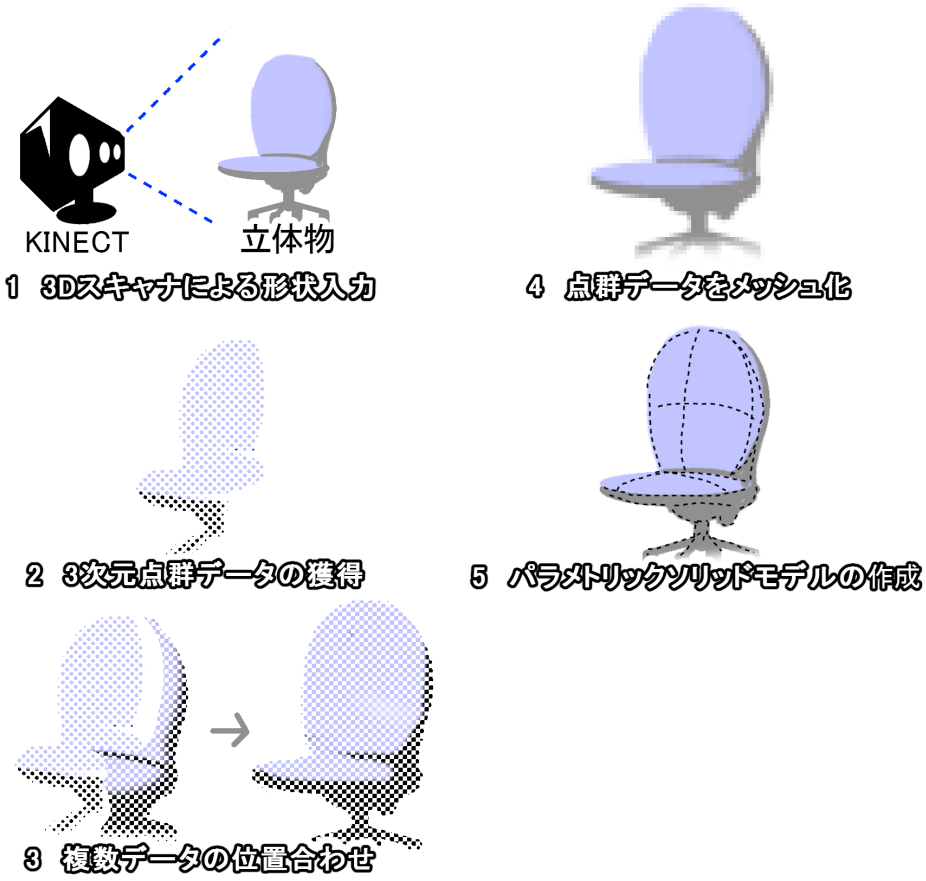

図 4　3 次元形状の入力から 3D モデリングまでのプロセス

3 章　Kinect による立体形状の取得

3.1　Kinect について

Kinect for Xbox 360 は、2010 年にマイクロソフトが製品化した Xbox 360 向けのゲームデバイスである。なお、2012 年には Microsoft Windows 向けに Kinect for Windows が製品化されている。本研究で使用するのは Kinect for Xbox 360 である。それは、本研究を開始した時期が Kinect for Windows の発表前であったからである。

Kinect にはマイク、RGB カメラと、深度センサが備わっている。このうちの深度センサに搭載されている近赤外光カメラと RGB カメラにより人間の関節を認識することができ、体の動きに対応した操作を可能にしている。

本来、Xbox 360 用に販売された本製品であるが、マイクロソフトは PC でプログラミングが可能になる開発ツールである Kinect SDK を提供している。この中の lib ファイルを利用することで、C#や C++でプログラミングを行なうことが可能になっている。

深度センサにより、Kinect は安価な 3D スキャナとしての利用ができる。ただし、この深度センサを構成している近赤外光プロジェクタと近赤外光カメラは 4m ほどまでの距離しか深度を測ることができないので注意が必要である。

本開発の第一の主要な部分は、Kinect で取得したカラー画像とデプス画像を整列させ、3 次元点群データに変換することである。ここでは、立体物の形状を取得したデプス画像を、実空間での 3 次元座標に変換する方法と、デプス画像とカラー画像を統合させたことによる副作用を述べる。

3.2　デプス画像から 3 次元点群データへの変換

Kinect の深度センサの出力は画像データ、すなわちデプス画像である。デプス画像は 640×480 画素で、各画素 13 ビットの精度を持つ。画素の数値は深度センサから被写体表面までの奥行きを mm 単位で表したものである。画像データの各画素は被写体上の 1 点に対応している。したがって、深度センサを中心とする 3 次元座標系において水平方向（X 軸）、垂直方向（Y 軸）、奥行き方向（Z 軸）の座標値を計算することができる。このなかで、画素の数値は Z 軸の値に相当する。X 軸、Y 軸の座標値は、画像座標と深度センサ

の視野角および奥行き（Z 値）から、次のように求める。

Kinect の深度センサから得られるのは物体からセンサまでの奥行きである。センサから物体までの水平垂直位置は画素単位で取得する。画素単位の座標と Kinect の視野角、距離から、三角関数を用い水平垂直位置の実寸を求める。

被写体表面の点Pの、実空間での3次元座標を(X,Y,Z)とする。図5は、X軸に平行で点 P と投影中心(B)を通る平面と、近赤外カメラの投影空間が交わることで形成される三角形 ABC である。図で、角 ABD は近赤外カメラの水平視野角の半分であるから 28.5 度になる。wはデプス画像における中心から点 P までの X 軸方向の画素単位の距離である。三角関数を用いると、辺 AD の長さは Z・tan28.5 となる。この長さはデプス画像の解像度を640×480 画素とした場合、320 画素分に相当する。

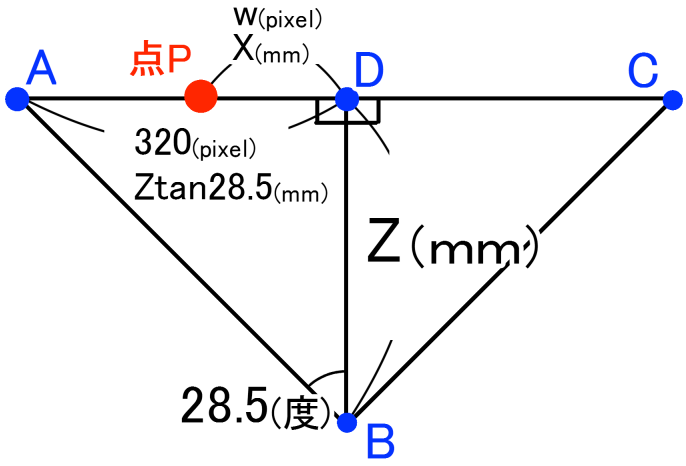

図 5　点 P の水平位置を示す図

従って、320:w=Z・tan28.5:X という比例式が成り立つ。これを変形することで X=(w・Z・tan28.5)/320 という式となり、デプス画像から得た位置 w と画素値 Z から点 P の実空間での 3 次元座標 X を求めることができる。

Y 座標においても同様に、w をデプス画像における中心から点 P までの Y 軸方向の画素単位の距離、角 ABD を垂直視野角の半分の 21.5 度、辺 AD の

画素単位での長さを 240 として計算すれば、点 P における 3 次元座標 Y を求めることができる。

3.3　カラー画像とデプス画像の整列

図 6 はセンサの搭載位置を表している。RGB カメラと深度センサは搭載されている位置がずれているため、視差による取得データのずれが生じる。よって対応する座標位置を合わせる必要がある。

KinectSDK にある NuiImageGetColorPixelCoordinatesFromDepthPixel 関数がデプス画像に対応するカラー画像の座標位置を計算する。

図 6　カメラの搭載位置

図 7 は座標位置を合わせる前のデプス画像とカラー画像の取得画面、図 8 は座標位置を合わせた後のデプス画像とカラー画像の取得画面である。このデプス画像において、黒部分（画素値 0）は距離が計測できなかった部分である。白くなるにつれてセンサと位置が近くなるように表示した。図 7 の赤丸の部分はデプス画像にはなく、カラー画像のみが取得できている範囲で、青丸はデプス画像のみが取得できている範囲。図□ではカラー画像の位置が右上に移動し、デプス画像と対応付けることができた。しかし、もともとデプス画像はカラー画像より下の範囲に取得範囲を持つため、図 7 のカラー画像の下方でデータが欠けてしまう。

図 7　座標合わせを行う前の入力画像

図 8　座標合わせを行った後の画像

4 章　PCL による 3 次元点群データの処理

4.1　PCL について

PCL(Point Cloud Library)は 3 次元点群データを扱うことのできるオープンソースソフトウェアである。3 次元点群とは、3 次元座標の集合である。PCL にはフィルタリング、特徴推定、サーフェス再構成、位置合わせ、モデルフィッティング、セグメンテーションなど、たくさんの最先端のアルゴリズムが含まれている。

今回はそれらのうち、はずれ値フィルタリング、ダウンサンプリングフィルタリング、法線ベクトルの推定、特徴推定、3 次元点群データの初期位置合わせ、3 次元点群データの詳細な位置合わせを用いた。

ここからは PCL を用いた 3 次元点群データのフィルタリングと位置合わせについて記述する。なお、PCL を利用する前に、Kinect によって取得したデータを PCL に対応するファイル(pcd 形式)に変換し、それを読み込み直すことによって、Kinect のデータをそのまま PCL で扱えるようにした。

4.2　はずれ値フィルタ

Kinectで取得した3次元点群データにはノイズ(位置の誤差)が含まれる。このノイズの原因は、測定面の反射率が低いことによって測定値の信頼性が悪くなっている場合や、物体の端面（エッジ）を測定しことによる誤計測などが考えられる。このようなものの中で、大きな誤差や、完全に誤った計測値をはずれ値とよぶ。はずれ値が存在すると、後述する特徴量の推定が難しくなり、3 次元点群データの位置合わせが困難になる。よって、このはずれ値を除去するフィルタリングを行う必要がある。はずれ値フィルタは次のように動作する。

入力データの全点について、それぞれの点から複数の近傍点までの平均距離を計算する。平均距離の分布を正規分布と仮定する。その分布に対する標準偏差をσとするとき、分布の平均からσ以上はなれた距離をもつ点を取り除くことではずれ値を除去する。今回は、近傍点の数を 50 点とした。

図 9 は、はずれ値フィルタを行う前の 3 次元点群データ、図 10 は、はずれ値フィルタを行った後である。図 9 と比べて、図 10 は後方の壁の点群がかなり削除されている。これは、センサから対象の立体物が遠くなるほどはずれ値を多く含んでいることを示している。この例では点の個数は 26 万

個から 23 万個まで減っている。

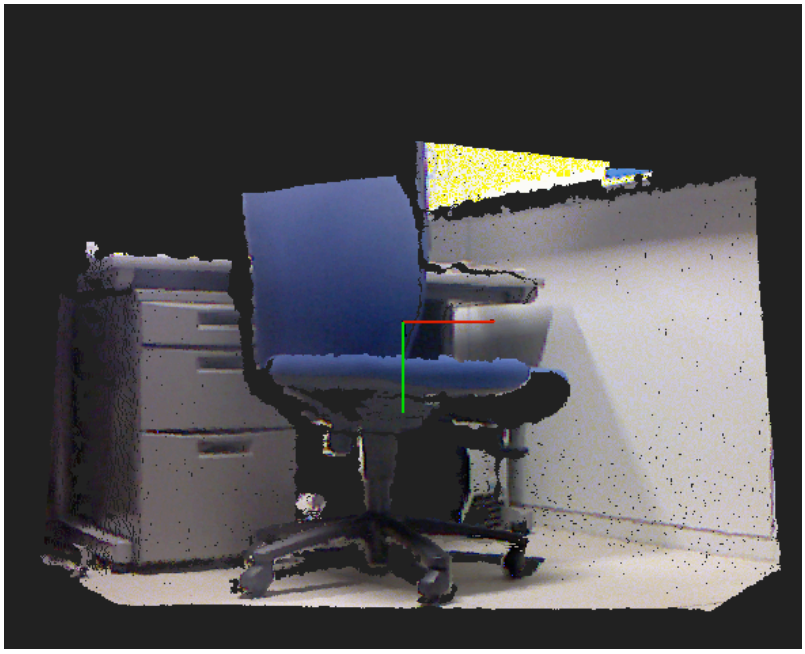

図 9　フィルタリング前の 3 次元点群データ

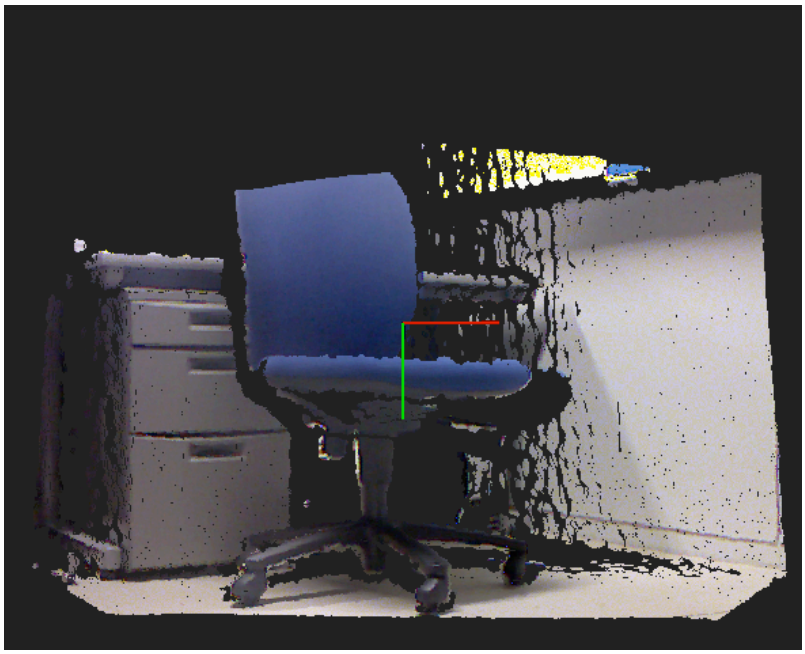

図 10　はずれ値フィルタを処理した 3 次元点群データ

4.3 ボクセルグリッドフィルタ

Kinect は 640×480 画素の点群を取得する。これだけのデータを処理する

必要がない場合には、点群の密度をさげるためのフィルタリングを行う。点群の密度を下げることにより、点群データの処理は高速になる。PCL には点群の密度を下げるダウンサンプリングフィルタの一つに、ボクセルグリッドフィルタがある。

図 11 はボクセルグリッドフィルタの原理をあらわしている。立方体のグリッドを配置し、そのグリッドの内の点の重心を求め、1 点に置き換えることで点の数を減らす。このとき、ボクセルの大きさを変更することで点の密度を調整できる。

図 12 は図 10 にボクセルグリッドフィルタを行った後の図である。大幅に点が間引かれている。この図の例では、点の個数が 23 万個から 1 万個程度に減った。

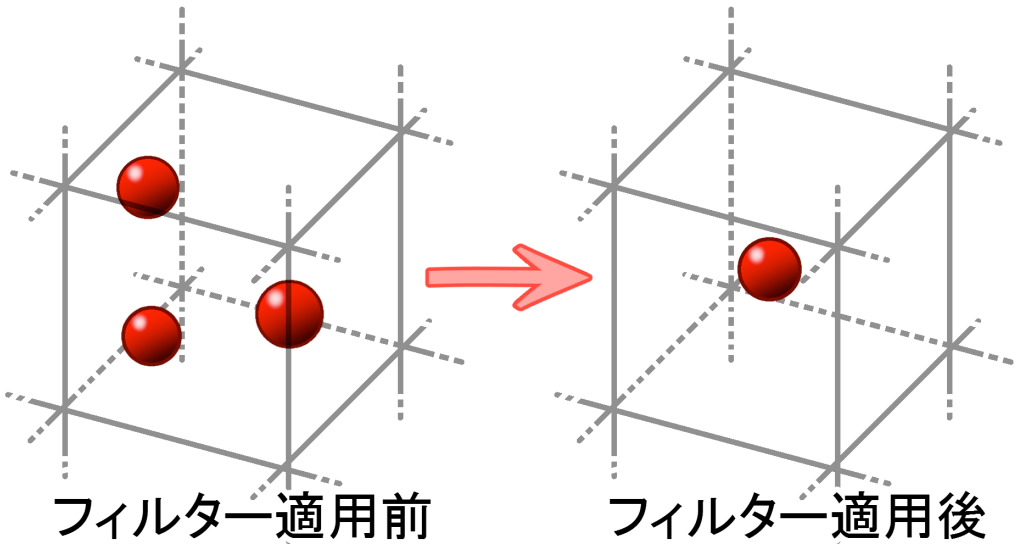

図 11　ボクセルグリッドフィルタによる置換

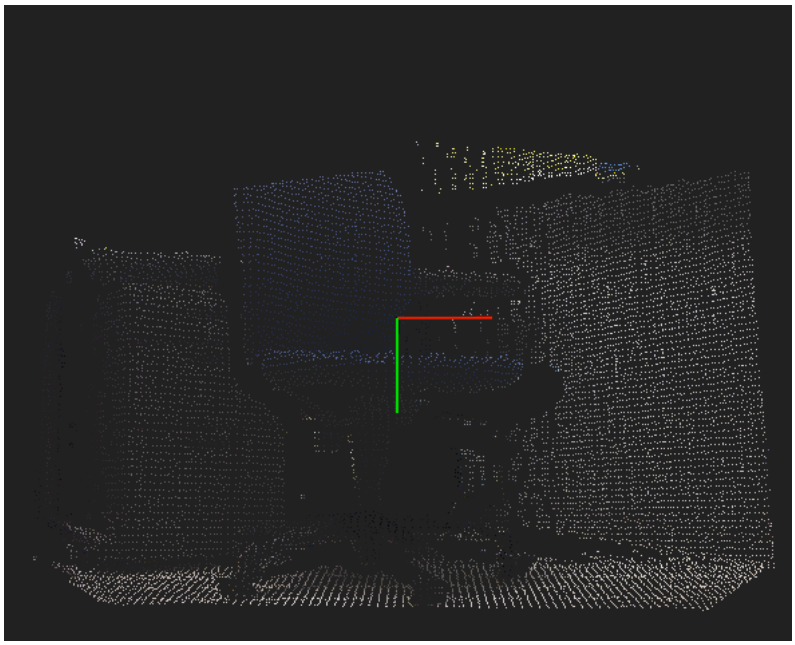

図 12　図 10 のデータにボクセルグリッドフィルタを処理した 3 次元点群データ

4.4　初期位置合わせ

Kinect による 1 回の測定で取得される 3 次元データは、測定対象物の一部分の形状を表しているに過ぎない。立体物の完全な 3 次元データを取得するためには、複数の視点から取得した 3 次元点群データを一つに統合する必要がある。複数の点群データを、お互いに独立して取得した場合、それらの位置関係を事前に知ることができない。このとき、まず、複数の点群データを粗く位置合わせし、その後、ICP アルゴリズムで高精度に位置合わせを行う。

ICP アルゴリムは、繰り返し的な処理で位置合わせを高精度化する方法である。そのため、初期状態において 2 つの 3 次元点群データの位置が大体あっていないと有効に働かない。初期位置合わせを行うために、SAC-IA(Sample Consensus Initial Alignment)関数を用いる。

SAC-IA は初期位置合わせのために RANSAC（Randum Sample Consensus）を用いる方法である。位置合わせを行う 2 つの 3 次元点群データから、それぞれ複数の特徴点を抽出する。ここで、点群データにおける特徴点とは 3 次元形状の特徴（コーナーなど）を表すことができる点のことである。2 つの点群データの一方をターゲット、他方をインプットとして、特徴点間でランダムに対応付けをする。対応付けた特徴点間が最も近づく

ように座標変換を行う。そのときに生じる位置の差を、特徴点の誤差とする。この対応付けを一定回数行い、最も2つの特徴点の誤差が低くなった場合を座標変換として採用する。そしてターゲットに合わせてインプットを回転、拡大縮小、平行移動をおこない、再配置することで、大まかな位置を合わせることができる。

SAC-IA は点群の特徴点からそれらを整列するので、まず FPFH(Fast Point Feature Histograms)で特徴点を求める。さらに FPFH を使用するときに法線ベクトルも必要となる。よって位置合わせは、法線ベクトルの推定→特徴点の推定→SAC-IA の実行→ICP アルゴリズムの実行という手順を踏む。

図 13 は 2 つの異なる測定位置から立体物の計測を行った図である。視点 1 では立体物の上面、視点 2 では立体物の側面を計測した。図 14 は、図 13 の測定結果から 2 つの 3 次元点群データを求め、初期位置合わせを行わず ICP アルゴリズムによる精密位置合わせを行ったものである。2 つの点群データの測定位置が大きく違うため、正しい位置に収束していない。

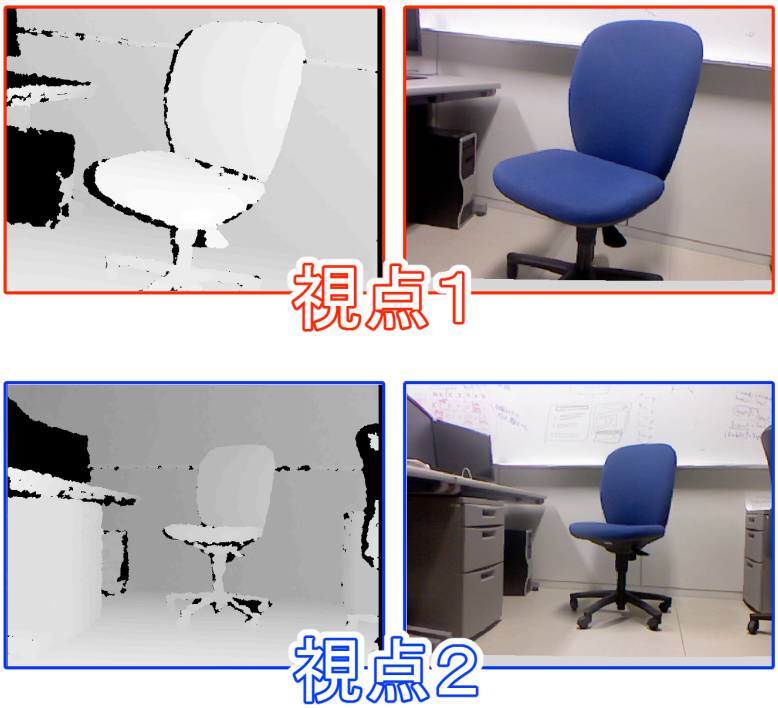

図 13　2 つの異なる測定位置から立体を計測した図

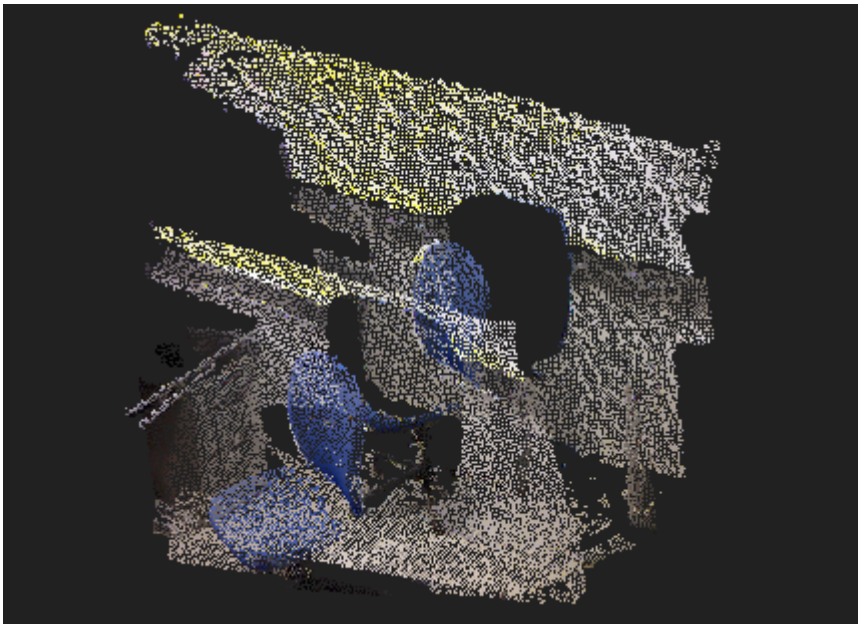

図 14　ICPアルゴリズムのみで位置合わせを行って、点の統合をした場合

図 15 は、測定位置の異なる 2 つの点群データを、初期位置合わせのみを行ったものである。図 14 と比べると対応付けた特徴点が近づいた。

図 16 は図 15 を他の角度から見た図である。特徴点の距離は近づいたが、大きなずれが生じてしまっている。

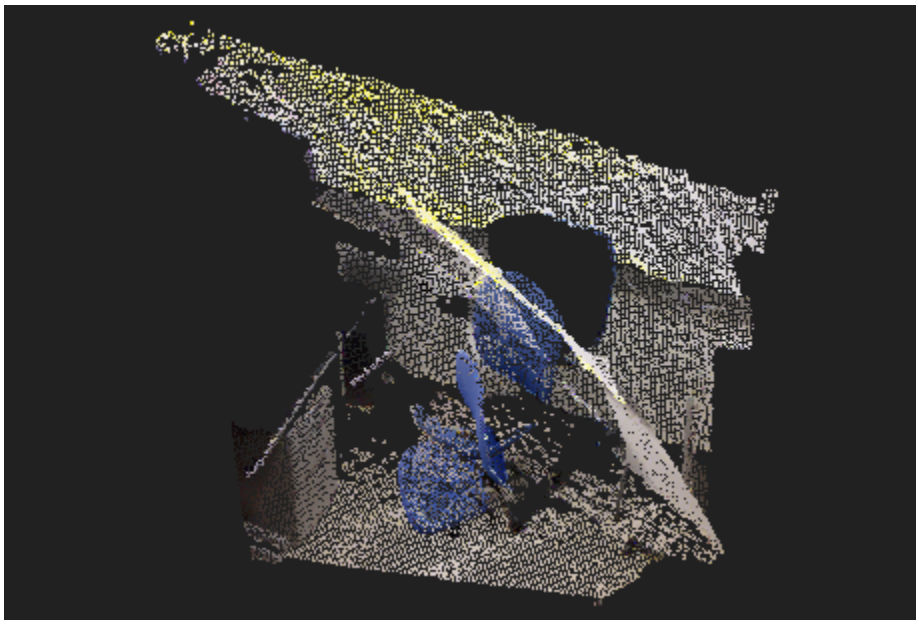

図 15　初期位置合わせを行って、点の統合をした場合

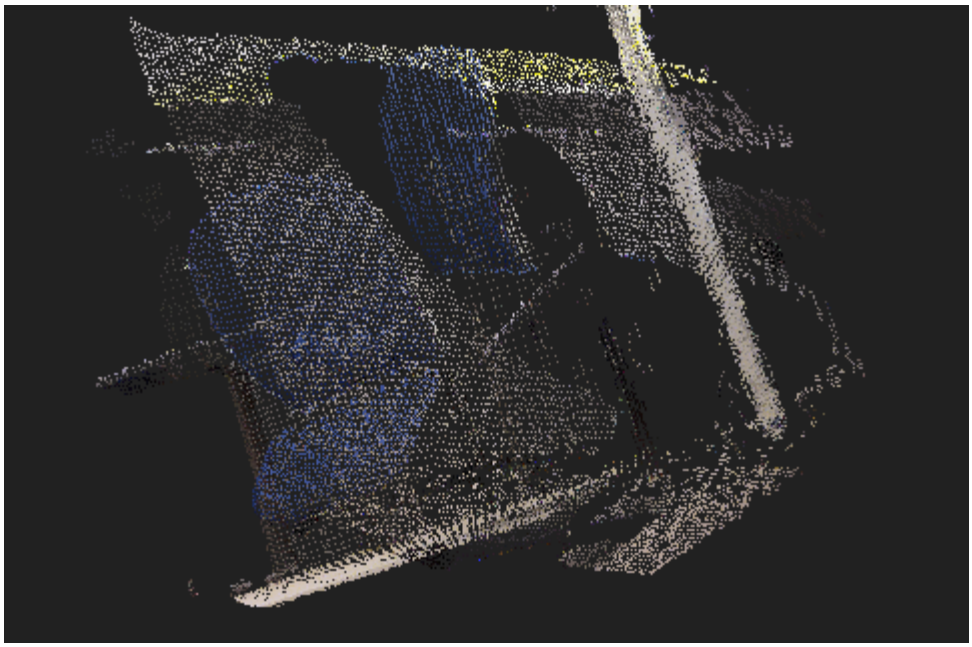

図 16　初期位置合わせのみの場合のずれ

4.5　ICPアルゴリズムを用いた高精度位置合わせ

ICPアルゴリズムは3次元点群データ間の位置合わせを行う手法である。ICPアルゴリズムは6つの段階で処理を行う。第一に位置合わせを行う2つの3次元点群データのうち一方から点を選択する。第二に選択した点と他方の点との対応を求める。第三に対応した点の組に重みをつける。第四に不要な点の対応を除外する。第五に誤差量を求める。第六に誤差量を最小にする。

ICP アルゴリズムの計算過程で、対応させる点を間違えた場合、正しく収束しない。従って、初期位置合わせが重要である。

ICPアルゴリズムを行ったあと、2 つの 3 次元点群データを合成してひとつにする。このときデータが膨大になることを防ぐため、再びボクセルグリッドフィルタリングを行う。これを繰り返し、複数の視点から得た3次元点群データを合成する。

図 17 は図 15 に ICP アルゴリズムを用いて高精度な位置合わせを行い、統合したものである。図 15 と比べてよりデータが一致した。

図18は図16と同じような角度で図□□を見た図である。図□□にあった大きなずれがなくなった。

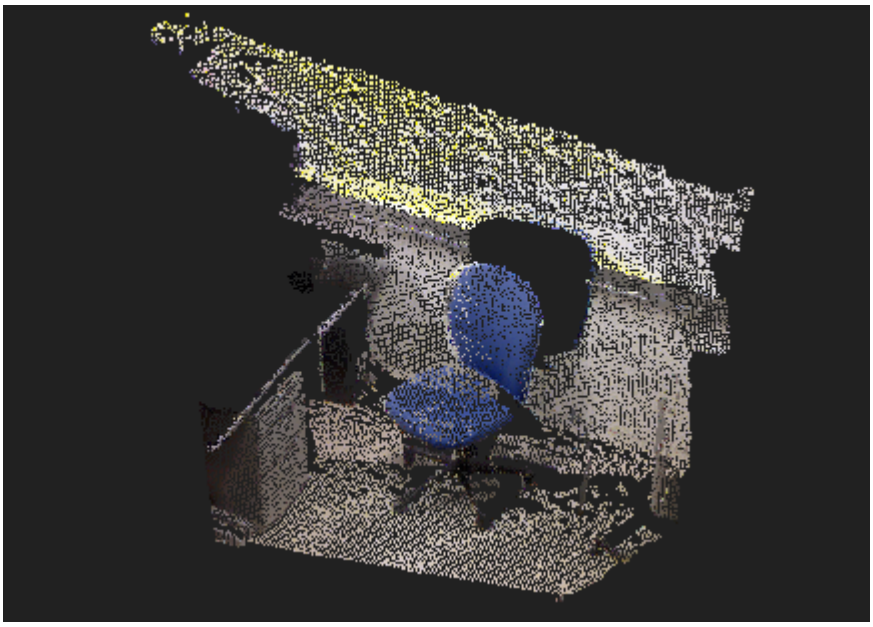

図 17　初期位置合わせ後、高精度な位置合わせを行った場合

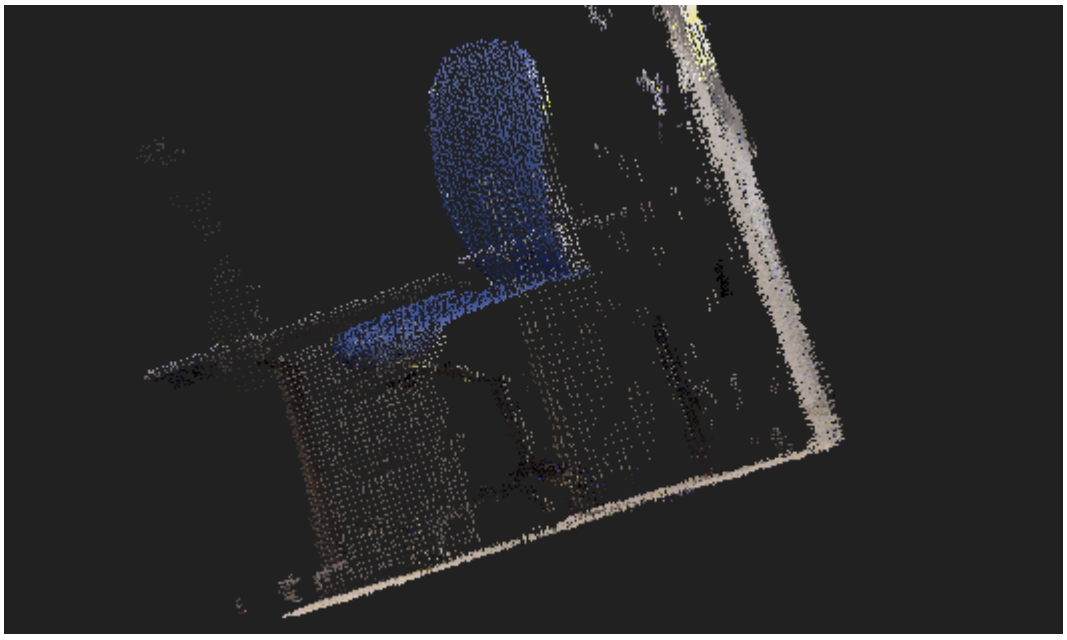

図 18　初期位置合わせ後、高精度な位置合わせを行った場合のずれ

5 章　結論

5.1　成果

本研究ではKinectとPCLを用いて 3Dモデルを作成するために、立体物の 3 次元形状を計測し、獲得した 3 次元点群データを処理するプログラムを開発した。

3Dスキャナからモデリングを行うためには、最初にスキャナから 3 次元点郡データを取得する。次に複数視点からの 3 次元点群データの位置合わせを行う。取得した 3 次元点群データにメッシュを張り、パラメトリックソリッドモデルに加工する必要がある。開発したプログラムでは、この工程のうち位置合わせまでを行えるようにした。

Kinectを 3Dスキャナとして使用するため、カラー画像とデプス画像から 3 次元点群データを獲得した。このとき、実空間での水平垂直位置を三角関数によって求めた。また、カラー画像とデプス画像の取得範囲が違うため、3 次元点群データは、デプス画像とカラー画像の取得範囲が被っている部分のみ取得できた。

3 次元点群処理では、はずれ値フィルタリングでノイズを除去し、ボクセルグリッドフィルタリングでデータ数を低減した。ボクセルグリッドフィルタによるダウンサンプリングを行うことで、その後の処理が高速化した。次に、フィルタリングを行った後の 3 次元点群データを 2 つ用意し、位置合わせを行った。このとき、SAC-IAによって初期位置合わせをすることで対応する点の距離を近づけ、ICPアルゴリズムによって高精度な位置合わせを行った。

KinectとPCLを用いれば簡単に入力から複数視点から見た 3 次元点群データの統合まで行えることがわかった。

5.2　課題

実際に立体物の測定を行うために Kinect の評価と実用的なインタフェースを実装する必要がある。現段階では別々にデータを入力して処理を行っているため手間がかかる。

モデリングに関しては位置合わせまでしか行うことができなかった。最終的にはメッシュを張り、パラメトリックソリッドモデルの作成まで行う必要がある。また、そのために必要な立体物の切り出しを行うフィルタリ

ングを実装しなければならない。

位置合わせに関しても、2つの異なる点から観測した3次元点群データを位置合わせし、統合する場合しか行っていない。多くの3次元点群データで次々と位置合わせをおこなった場合の試行をしなければならない。

謝辞

本研究に察して、様々なご指導を頂きました蚊野浩先生に感謝の意を表します。

参考文献

[1]DERiVE, 2012 年<http://derivecv.tumblr.com/>
（アクセス：2013 年 1 月 4 日）
[2]谷尻豊寿, 「KINECT センサー　画像処理プログラミング」,
カットシステム, 2011 年.

付録

[1]今回開発したプログラムにおける PCL のパラメータ

※設定してない値はデフォルトのまま。

はずれ値フィルタ	
近傍点の個数	50 個
基準となる標準偏差の倍率	1 倍
ボクセルグリッドフィルタ	
ボクセルサイズ	x,y,z いずれも 0.15m
法線推定	
検索する半径	0.03m
特徴点の推定	
検索する半径	0.08m
SAC-IA	
対応点の組数	5 組
2 点間の対応づけを行う最大値	1m
SAC-IA の実行回数	50 回
ICP アルゴリズム	
RANSAC を行う際の除去範囲	0.1m
2 点間の対応付けを行う際の最大値	0.1m
ICP アルゴリズムの実行回数	100 回

[2]主要なプログラム

KinectSample2	
CkinectViewer.cpp	Kinect の動作を制御。カラー画像とデプス画像を取得し、3 次元点群データに変換する
KinectSample2.cpp	アプリケーションのクラス動作を定義
KinectSample2Dlg.cpp	カラー画像とデプス画像の取得画面を表示する
PCLFirstTime	
PCLFirstTime.cpp	PCL で 3 次元点群データの処理と 3 次元点群データの表示を行う