

3D-Hough法を用いた屋内空間の三次元平面形状の検出

安齋, 達也 / ANZAI, Tatsuya

(発行年 / Year)

2013-03-24

(学位授与年月日 / Date of Granted)

2013-03-24

(学位名 / Degree Name)

修士(理学)

(学位授与機関 / Degree Grantor)

法政大学 (Hosei University)

2012 年度修士論文

3D-Hough 法を用いた屋内空間の三次元平面形状の検出

3D shape detection of the indoor space based on

3D-Hough method

指導教員 藤田 悟教授

法政大学大学院

情報科学研究科 情報科学専攻

11T0003

アンザイ タツヤ
安齋 達也

Tatsuya Anzai

E-mail: 11t0003@cis.k.hosei.ac.jp

Abstract

This paper describes methods for detecting the 3D shapes of the indoor space that is represented as a combination of planes such as a wall, desk, or whatnot. Detecting the planes makes it possible to perform calibration of multiple sensors and 3D mapping, and then produces various services such as the acquisition of life logs, AR interaction, and invader detection. This paper proposes and verifies three algorithms. First, it mentions a way to use 2D-Hough. The proposed technique converts 3D data into 2D data by slicing. Second, it mentions a way to use 3D-Hough. The proposed technique requires appropriate parameters with which performances are well controlled. Finally, it mentions another way to use 3D-Hough in the voxel space. The voxel space expresses more precise surface with cumulative 3D data, and makes it possible to detect the planes more accurately. This paper solves to detect the planes in those methods, detect intersection point formed by three planes in real time, and detect more than ten planes in reconstruction 3D space using 3D-Hough in voxel space.

目次

1. はじめに.....	1
2. 関連研究.....	2
2.1. 平面抽出法	2
2.1.1. 2D-Hough	2
2.1.2. 3D-Hough	3
2.1.3. 最小二乗法.....	4
2.2. RGB-D 情報を用いたマッピング	4
2.2.1. RGBD データを用いた ICP.....	4
2.2.2. Loop Closure Detection	5
2.3. Kinect	5
2.4. Kinect Fusion.....	5
2.4.1. GP-GPU.....	6
2.4.2. 符号付き距離関数.....	6
3. 2D-Hough による平面認識	7
3.1. 提案手法	7
3.2. 交線の抽出	7
3.3. 実験, 考察	8
4. 素データを用いた 3D-Hough による平面認識	10
4.1. 提案手法	10
4.2. 3D-Hough の高速化手法	10
4.3. 実験, 考察	10
4.3.1. 単数深度画像における平面取得精度, 時間検証	11
4.3.2. 深度映像における平面取得精度検証	13
4.3.3. 複数枚深度画像における平面取得精度, 時間検証	13
5. ボクセルを用いた 3D-Hough による平面認識	16
5.1. 提案手法.....	16

5.2. 実験, 考察	18
5.2.1. 平面取得速度実験.....	19
5.2.2. 多数平面取得精度実験	21
5.2.3. 角度間隔変化による平面取得精度実験.....	24
5.2.4. 点群量変化による平面取得精度実験	25
5.2.5. 近傍点変化による平面取得精度実験	27
5.2.6. 平面パラメータ分割アルゴリズムを用いた複数枚平面取得.....	30
6. 考察.....	38
7. おわりに.....	39
謝辞.....	39
参考文献	39

1. はじめに

近年、コンピュータは今まで以上に高速で演算処理を行えるようになり、三次元データを容易に扱えるようになった。物体の 3D オブジェクトから、デジタルデータ化された歴史的資産である建築物、都市全体のマッピングデータ等、大小様々な三次元データがそれぞれに適したサービス展開を繰り広げている。実在する物体の三次元データは手動で作成することも可能であるが、大量のデータを作成する際等にはセンサを用いた自動取得を行う技術が必要である。センサの種類には様々あるが、一般的に奥行きを取得するレーザセンサ、取り分け一視点ではなく画像センサの様にマップで取得可能なレーザセンサは高価であった。2010 年に Microsoft から発売された Kinect は安価でありかつ RGB データと深度データを同時取得出来るセンサである。Kinect の発売をきっかけに、深度センサを用いた研究や技術開発が盛んに行われるようになった。

本論文ではこの Kinect を使用し、屋内空間のモデリングを行うため、センサを用いて三次元空間に存在する特徴点である壁や机などの面を検知することを目的としている。空間内に存在する特徴を検知することによって、複数台センサのキャリブレーションやマッピングの補助を行うことを可能とする。三次元空間を正確に認識することによって、その空間内で様々なサービスを提供することが可能になる。例えば、ライフログの取得や AR インタラクション、セキュリティなどが挙げられる。本論文では取得した三次元情報である点群データからの平面形状抽出法を提案し、実装した。またその実行時間、取得精度についてパラメータを変化させながら実験を行った。

以下、2 節では本研究の関連する研究分野について述べる。3 節では 2D-Hough を用いた平面取得法について提案し実験を行う。4 節では 3D-Hough を用いた平面取得法について提案し実験を行う。5 節ではボクセル空間と符号付距離関数、3D-Hough を組み合わせた平面取得法について提案し実験を行い、6 節で考察を述べた後に 7 節にまとめを述べる。

2. 関連研究

2.1. 平面抽出法

2.1.1. 2D-Hough

一般的な 2D-Hough は画像処理に用いられる特徴抽出法である．空間中に存在する点群一つ一つに対してその点を通る直線への距離のパラメータを記録し，もっとも頻度の高かったパラメータを持つ線を直線として抽出する手法である．ここで，パラメータの取り得る値を有限個にするためにハフ空間を用いる．ハフ空間では一つの直線を原点から対象の直線に引いた法線の長さ ρ と角度 θ の 2 つのパラメータで表す．点 (x,y) を通る直線の式は下記の式(1)の様に表すことが出来る．

$$\rho = x \cos \theta + y \sin \theta \quad (1)$$

・パラメータを取り違える現象

パラメータを取得する際に，直線の誤検知が発生する可能性がある．誤検知を起こしてしまう理由としては，データにノイズデータが多いことと，直線の数が多く複雑化していること，取得する直線が短いこと等が例として挙げられる．

・複雑なデータ

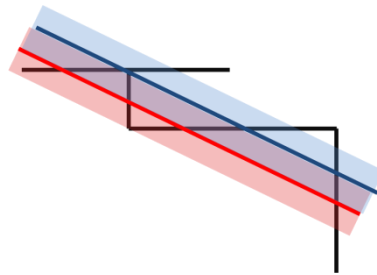


図 1. 複雑な点群データ

図 1 は複数の直線が存在する空間の図である．水平，垂直で複数本ある黒い線がデータ内の平面で，斜線である赤い線，青い線が誤検知した直線，及びその近傍である．中央部分の入り組んだ逆コの字型をなす直線部分は複雑で，データが精密であればパラメータを取り違えることはないがノイズの多い点群から直線検出を行おうとすると，赤い線，青い線上に存在する点群が本来の直線よりも多くなってしまい，誤検知を起こしてしまう可能性がある．

・広範囲のデータ



図 2. 広範囲の点群データ

図 2 は長い直線を誤検知してしまった時の図である．黒い線が本来の直線で，青い線がハフ変換によって誤検知してしまった直線を表している．点群データのノイズの乗り方や，センサによって取得出来るデータにひずみがある場合は一本の直線を複数本の直線に誤検知してしまう現象が存在する．

2.1.2. 3D-Hough

3D-Hough[1]とは一般的な Hough 変換を三次元空間上で行い，面の判定が可能になる様に拡張した技術である．三次元空間中の点 (x_p, y_p, z_p) を含む平面は，原点からの距離 ρ と，二つの角度 θ (方位角)と ϕ (仰角)で表すことが出来る．座標空間中の平面への距離 ρ は式(2)の様に表すことが出来る．

$$\begin{cases} x_p = \rho \sin \phi \cos \theta \\ y_p = \rho \sin \phi \sin \theta \\ z_p = \rho \cos \phi \end{cases} \quad (2)$$

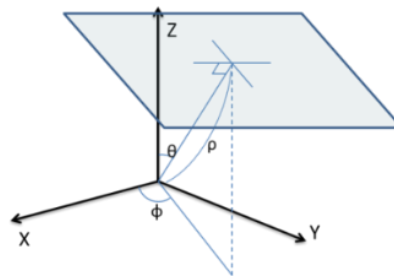


図 3. 座標空間内の平面のパラメータ

点群データはパラメータ領域上に頻度を情報として展開し蓄積され，検出頻度の高い順に面として検知する．

3D-Hough を利用した研究として航空機に搭載されたレーザスキャナによって都市の三次元モデルの形成を扱っている研究がある[2]．その研究では，屋根面の輪郭を正確に決定することで，建物の壁面形状を正確に捉えてより正確な建築物の形状を取得する．屋根がなす平面の抽出には 3D-Hough を用いる．この際，複数枚の面を一枚の面として認識してしまう事でデータの取り違いが起きる可能性がある．このような現象が起きないようにするため，取得した画像内から輪郭のエッジを取得し，セグメント化を行い，セグメントごとに 3D-Hough を行うことで解決している．

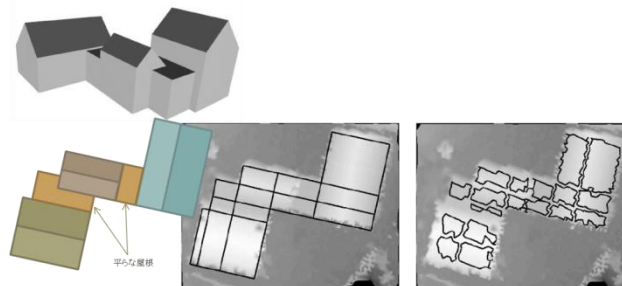


図 1. エッジ取得によるセグメント化

2.1.3. 最小二乗法

取り出した三次元情報付き画像を格子状に切り分け，各格子内で最小二乗を行い，平面方程式を決定する技術である．隣接した格子内の平面の傾きと比較し，同様のベクトルを持つと認識された場合は面の統合を行う．格子内に複数の向きの面があり，誤値と思われるベクトルが得られた場合は格子内で再分割を行い，ベクトルを再取得する．

最小二乗法により，平面を決定する計算は次のように行う．まず，平面の方程式を式(3)に示す．

$$ax + by + c = d \quad (3)$$

この平面と点 (x_1, y_1, z_1) の距離 r を式(4)に示す．

$$r = \frac{|d - (ax_1 + by_1 + cz_1)|}{\sqrt{a^2 + b^2 + c^2}} \quad (4)$$

複数の観測点に対して，この r の二乗の和を求め， $\frac{\partial \sum r^2}{\partial a} = \frac{\partial \sum r^2}{\partial b} = \frac{\partial \sum r^2}{\partial c} = 0$ となる a, b, c を求めることで平面の取得を行うことが可能である．

2.2. RGB-D 情報を用いたマッピング

RGB-D 画像から室内空間の 3D モデリングを行うためのマッピング技術である．RGB データのみを用いたフレーム間の位置推定は多数の研究がされているが，光の当たりかたで色情報が変化することで位置推定に誤差が発生する事象や，色情報の取得できない暗闇での位置推定精度の低下がネックになる．そこでこの手法では色情報と深度情報を組み合わせて，センサを持って巡回した経路に沿った 3D マップの自動生成を行う．この技術では色画像と深度画像を用いてカメラ視点の位置同定を行う RGBD データを用いた ICP と，既知の空間に帰ってきた際に位置の再調整を行う Loop Closure Detection の機構の二つに分けられる．機構の流れを下記の図 5 に示す．

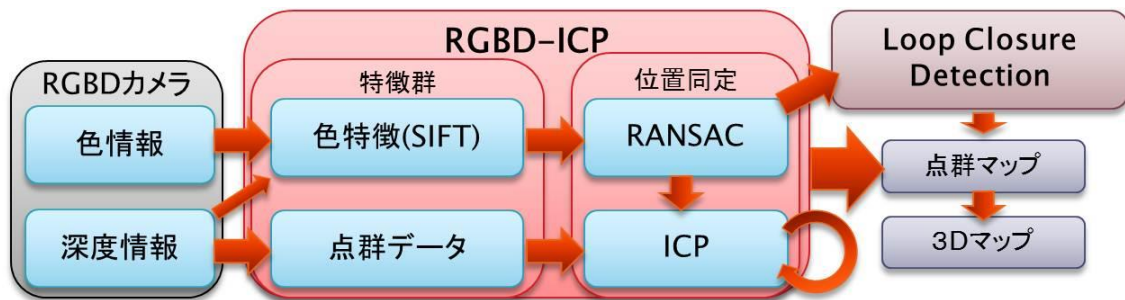


図 5. アルゴリズム

2.2.1. RGBD データを用いた ICP

RGBD データから特徴群を取り出し，連続するフレーム同士の位置同定を行う．既存技術である SIFT(画像データから特徴群の検出を行う技術)，RANSAC(2 つのデータを比較し，特徴のペアを決定する技術)，ICP(2 つの 3D データを比較し，関連付けされた任意の 2 点の距離誤差を最小にするためのアルゴリズム)を組み合わせることでこのアルゴリズムを構成する．

2枚の連続したフレームからカメラ位置がどのように変化したかをトラッキングするために、視点変換式を求める。まず2枚のフレームから SIFT を用いて RGB データ内の色特徴を抽出する。取り出した色特徴から RANSAC を用いて位置同定を行い、これを初期の視点変換式とする。求めた視点変換式を用いて2枚のフレームの 3D データ間の ICP を複数回を行い、漸近的に最適解に近づけてゆく。

2.2.2. Loop Closure Detection

ループクロージャ問題(Loop Closure problem)とは、位置推定をしながら移動しているセンサが大きな移動の後に元の位置に戻った際、長い時間をかけて誤差が累積して位置ずれを起こしてしまう現象のことである。本手法では一定間隔で空間の特徴を保持し、既知の特徴量を発見した際には同じ位置に戻ってきたと判別し、位置の最適化を行うことで改善を図る。特徴は 2.2.1.節で述べた SIFT による色情報を用いている。

2.3. Kinect

Kinect は Xbox360 向けに開発されたゲームデバイスである。複数のセンサを搭載していて、物理コントローラを用いずにジェスチャーだけでゲーム操作を行うための機器である。RGB カメラに加えて深度情報を取得する Depth カメラを搭載していて、 640×480 の解像度で RGB-D データを容易に取得することが出来る。現在は PC 向けに開発された Kinect for Windows も発売されている。

深度画像の取得可能範囲は 50cm から 10m とされていて、空間内に存在する人物の位置を取得しモーションキャプチャを行うことが可能である。人物の骨格モデルを用いて画像内にいる複数人の位置情報やジェスチャー判別を行うことが出来る。これらの技術は、Kinect の深度カメラを提供している Natural Interaction のためのフレームワークである OpenNI や、Microsoft 社による開発キット、Kinect for Windows SDK 等を用いて PC 上で操作することが可能になる。

Kinect では、 640×480 ピクセルの色画像のピクセル一つ一つに距離データが付加されている。よって最大約 30 万個の位置情報データを 1 フレームごとに取得することが出来る。このデータから点群は Kinect を原点にした三次元座標に変換ことが出来る。しかし、得られる映像には多くのノイズや欠落が発生する。

2.4. Kinect Fusion

Kinect Fusion[3][4]は Microsoft Research らが開発したリアルタイムに周囲の環境を 3D モデル化する技術である。2.3 節で述べられている Kinect を手に持った状態で動かしながら複数視点からの映像を取得し、調解像度のデータを作成することでより正確な 3D モデルをリアルタイムで構築する事が可能になっている。Kinect Fusion は下記の動作の繰り返しにより 3D モデルを構築する。

1. Kinect から Depth データを取得し三次元座標に変換
2. ICP アルゴリズムを用いて Kinect の位置推定を行う
3. あらかじめ作成したボクセル空間上で
三次元モデルを作成
4. レイキャスティングによる画面への描画を行う

また再構成された環境に対して物理シミュレーションを行うことで、高度なインタラクションを実現している。

2.4.1. GP-GPU

GP-GPU とは本来画像処理を受け持つ GPU(Graphics Processing Unit)をそれ以外の目的に応用する技術である。GPU は大量の計算を高速で行うことに優れている。これを汎用的(General Purpose)に活用することで処理時間の短縮を可能にする。利用には専用のプログラムを書く必要があり、分岐条件などが原因で発生するオーバーヘッドが起こらないようにしなければならない。

2.4.2. 符号付き距離関数

符号付き距離関数とは直行格子系における形状表現法である。直行格子線上の交点にそれぞれ符号付の値(F)と信頼度を現す値(W)を保持させ、 F の値の正負によってその位置に物体があるのか無いのかを判断する。仮に F が負の値の時に物体があると仮定すると、物体の有無は下記の図 6 の様に表現することが可能である。

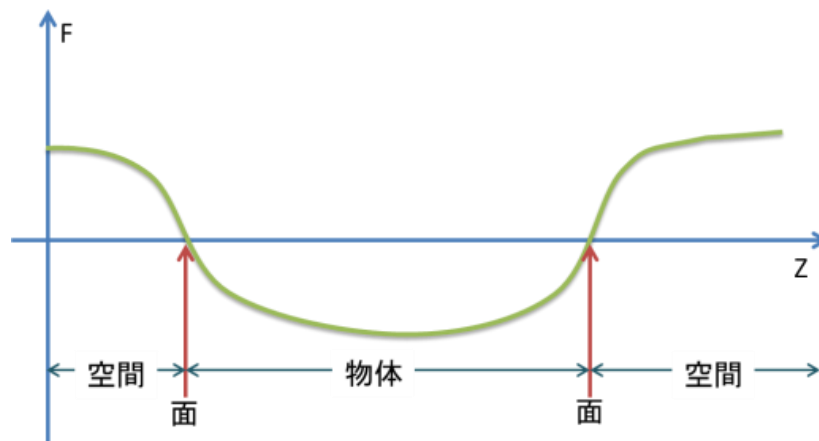


図 6. 符号付距離関数

この時 F の値が 0 になる部分を表面として検知することが可能である。直行格子線上の交点の各点から、前後左右手前奥の点と比較した際に符号の変化した位置が面の存在する場所となる。

3. 2D-Hough による平面認識

3.1. 提案手法

Kinect を用いて取得した深度付き画像から，空間内に存在する平面と平面，更に平面と平面がなす交線を特徴として抽出するために 2D-Hough を用いる．2D-Hough 変換は二次元空間で使用するので，取得出来る三次元データから二次元空間のデータセットに変換を行う．この変換については 3.1.1. で述べる．求め出した壁面と壁面が成す線の交点群から回帰分析によって壁面と壁面の交線を取得する[5]．

3.2. 交線の抽出

取得した深度情報付き画像を，2DHough 変換を行うために輪切りにしていく．壁に対して平行に切り出してしまうと，Kinect からの視点が仰俯角となっているため，x 軸に平行でかつ仰俯角を放射状に変化させる様に輪切りしていく．輪切りした平面内の深度 xy 空間の二次元座標として，確率的 Hough 変換を行っていく．

三次元空間上の点 (x,y,z) を二次元座標に変換する際の座標を式(5)に示す．

$$(x, \sqrt{z^2 + y^2}) \quad (5)$$

取り出された線分は最適化された後に近接する線分との交点となりえる点を計測し特徴点を抽出していく．この特徴点を繋いだ線分が壁と壁の交線になる．

二次元座標内で処理され抽出された点，線は三次元空間に再変換し，描画を行う． f を Kinect の焦点， YP をピクセル画像上の高さとした時，二次元座標の点 (X,Y) から三次元座標に変換した際の座標を式(6)に示す．

$$(X, Y \times \frac{YP}{\sqrt{f^2 + YP^2}}, Y \times \frac{f}{\sqrt{f^2 + YP^2}}) \quad (6)$$

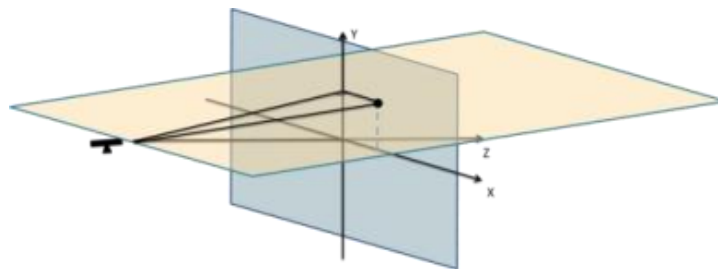


図 7. 三次元空間，二次元空間の関係図

Kinect から取得出来るデータの解像度の関係から，480 枚の輪切り画像を生成する．図 8 の取得する空間の元画像の上部に示されている直線部分で輪切りを行った画像が図 9 左になる．図内の点群を用いて Hough 変換を行った結果，この例では四本の直線を取得することが出来た(図 9 中央)．Kinect から取得することの出来る点群データにはばらつきが発生しているため，別の角度で取得した輪切り画像で Hough 変換を行った際には取得出来る直線にもばらつきが発生していた．直線と直線の交点は壁面と壁面のなす交点という特徴点となる．



図 8. 元画像

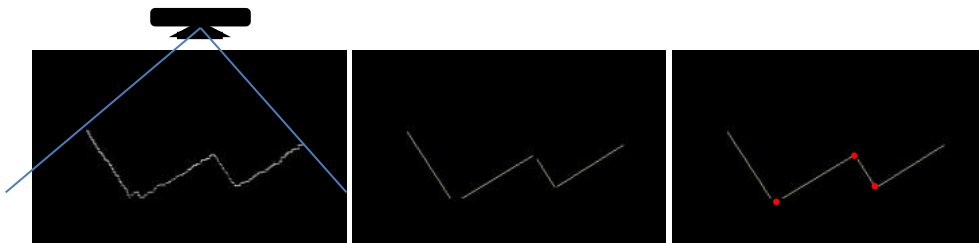


図 9. 輪切りした画像(図左), Hough 変換後の画像(図中), 直線の交点(図右)

3.3. 実験, 考察

Kinect から得られる三次元空間内に存在する平面と平面のなす交線の取得精度について検証を行う。取得した深度画像から、提案手法を用いて面、面と面の交点を取得し、実空間距離と比較を行う。今回は図 10 の黒枠で囲んだ部分に存在する 2 枚の壁面のなす交線の取得精度を測る。



図 10. 実験環境

2D-Hough によって取得した壁面をなす直線群が図 11 中の①となる。これらの直線の交点が②の点群となり、それらを回帰分析で求めた直線が③になる。この直線と実空間との距離の比較結果が下記の表 1 である。

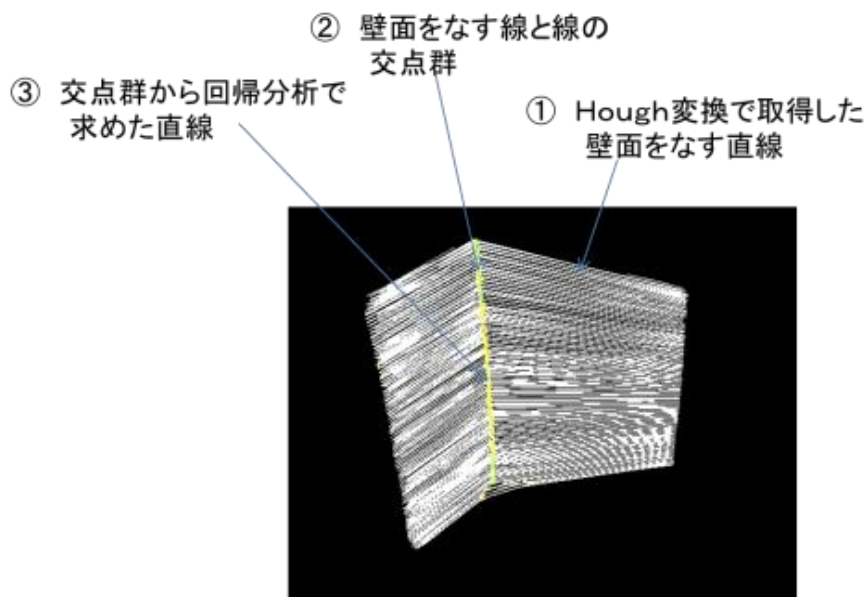


図 11. 実行結果

表 1. 取得したデータ

	直線への距離(cm)	線の高さ(cm)
実距離(cm)	220	155
取得した距離(cm)	219~223	159

図 11 より，壁面をなす平面は大まかには取得できているが，輪切りした空間ごとに取得出来た直線にばらつきが生じた．特に Kinect から距離の遠い部分(画像内の四隅等)の値には歪みが生じている．この誤差は，壁面と壁面の交線を求める際の回帰分析によってまるめ込まれているため，直線への距離や線の高さは表 1 の通り誤差数センチ以内で取得することが可能となっている．また空間内にある壁面の枚数が増えた状態で同様の実験を行った際には，一枚の面をなす点群量が少なくなってしまうため 2D-Hough した際に 2 枚の壁面を 1 枚の壁面と誤認識してしまう事象が発生した．パラメータ設定を変更し，2 枚を個別に認識する様にすると，今度はノイズデータのせいで一枚の壁面を 2 枚と認識してしまうことが起こった．

実験結果から，Kinect から取得出来るデータを輪切りにして，2D-Hough 変換を行うには Kinect から取得出来るデータ中のノイズ量が多いために直線検出に誤検知が発生しやすいことが判明した．本提案手法は 2 枚の壁面の交線を検出することは可能であるが，それ以上の複雑な面と面の交線，交点を取得することは難しいということが判明した．

4. 素データを用いた 3D-Hough による平面認識

4.1. 提案手法

2.1.2.節で述べた 3D-Hough を用いて Kinect から得られる三次元空間内にある平面の認識を行っていく。

まず、画像内にある全ての点に対して、その点を通る面のパラメータの検出頻度を保持していく。この時、面が取り得る角度パラメータをどのような間隔で取得していくかによって、頻度検出の時間と精度が変化する。面の取り得るパラメータは方位角が $0 \sim 360^\circ$ 、仰角が $0 \sim 180^\circ$ となる。これらの角度全ての組み合わせに対して、原点からの距離 ρ を求める。方位角 θ 、仰角 ϕ 、元の座標を (x, y, z) とすると、 ρ は式(7)の様に表すことが出来る。

$$\rho = x \cos \theta \sin \phi + y \sin \theta \sin \phi + z \cos \phi \quad (7)$$

全ての点を取り得るパラメータの頻度検出が終わったら、最も高い頻度で検出されたパラメータを面と認識する。この際、もう一度点群に対して検索を行い、その点の位置と取得した角度パラメータ θ, ϕ を式(7)に代入し、得られた結果とパラメータの距離を比較し、誤差が一定の値未満だった際にその点を頻度グラフから削除する。一定以上の枚数の面が見つかるか、その時点で最大頻度のパラメータの個数が一定未満になるまでこれを繰り返す[6]。

4.2. 3D-Hough の高速化手法

3D-Hough を行う際に、取得出来る全ての点に対して、より細かい間隔で角度のパラメータを収集することでより高い精度で面の検出を行うことが可能である。しかし、3D-Hough は計算量が多く、実行時間を大量に必要とする。

ここで Kinect を地面に水平に置いた際、取得データ内に存在する面の大半が壁面や床、天井など、Kinect から見て水平か、垂直であると仮定することが出来る。そこで仰角が $0 \sim 15^\circ$ 、 $75 \sim 105^\circ$ 、 $165 \sim 180^\circ$ となる面だけ計測を行う事で計算量を減らし、高速化することが可能である。

この手法は Kinect が地面に対して水平方向を向いていることが前提である。Kinect が向いている方向を推定することが可能であれば限定する角度を変更して高速化を図ることが可能であるが、Kinect が地面に対してどのような仰角方向を向いているか推定することが出来ない場合は利用できない。

4.3. 実験, 考察

3D-Hough 変換を用いて、室内空間の壁面を認識する精度や時間などについて検証する。

今回は Kinect から取得した深度画像の枚数、使用する点群のダウンサンプリング、取得する面のパラメータ(角度)の精度を変化させて実験を行う。

4.3.1. 単数深度画像における平面取得精度，時間検証

Kinect から取得した画像を一枚使用して平面の取得精度と時間の検証を行う．取得する壁面パラメータの角度(方位角，仰角)の精度を A として，値を変化させて取得することの壁面のパラメータについて検証を行う．この実験では，点群量のダウンサンプリング率は 1/16 とする．実験で得られた面の枚数は下記の図 12 の通り 3 枚である．

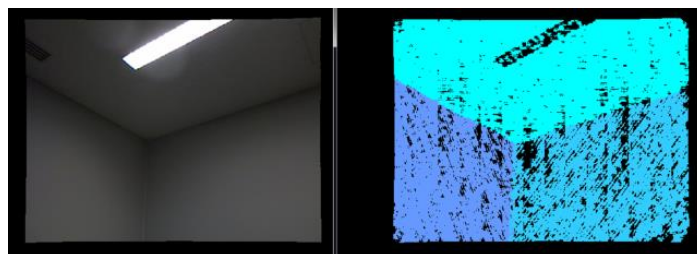


図 12. 生成した三次元空間

取得出来た面のうちの 1 枚，蛍光灯がある天井部分にあたる面のパラメータ取得精度についてまとめたものが表 2 である．

表 2. 単一視点，単一方位の精度実験

	A				
	0.2°	0.5°	1°	2°	5°
取得速度(秒)	1380	222	31	5.5	0.8
方位角 θ (度)	90.6	91	91	90	90
仰角 ϕ (度)	150.6	151	151	150	150
距離(cm)	186	185	185	188	188
取得した頻度	2621	2124	2124	1790	1790

角度の精度を変えることによって，取得精度の誤差が生じるものの，どの値においても壁面を正確に認識できているものであると考えられる．ここで発生する精度誤差は，大きい面を取得した際に面の一部(四隅等)をパラメータに含まれる面と検知することを難しくすることが起こる．平面に含まれる点群の閾値を大きくとることで点群の検知を行う事が出来るが，ノイズデータも同時に取得してしまう．

次に，同じデータを用いて角度 A と点群データ P の値を変化させてデータの取得速度についての実験を行う．

表 3. 単一視点，単一方位の速度実験

単位(秒)		P					
		1/1	1/4	1/16	1/64	1/256	1/400
A	0.5°	3762	958	245	69	25	18
	1°	557	148	38	8.3	2.9	2.2
	2°	87	23	5.6	1.7	0.6	0.5

P の値に関しては、 $1/256$ の点群量でも十分な面の取得が可能であった。この実験で取得出来た面の枚数に変化は無かったものの、 $1/400$ の点群量を用いた際には、一つの面の取得精度が多少低下した。よって実験結果より、点群量は $1/256$ より多くすることが望ましいと考えられる。

本実験では空間内に三枚の面がある空間で実験を行ったが、この実験空間に存在する面はすべてある程度の情報量を保持している。空間内にある面が持つ点群量が少ないと、ダウンサンプリングした際に面を発見することが難しくなる。下記の図は、点群量を $1/256$ 、パラメータの角度を 2 度間隔で面の取得を行った図である。空間には面が 5 枚あるため一枚一枚の面が持つ点群量は少なく、特に下記の図 13 内で赤枠で示している面の情報量は特に少ない。センサを動かして様々な位置から平面取得を行った際に、一番情報量のすくないこの面から検知することが出来なくなった。

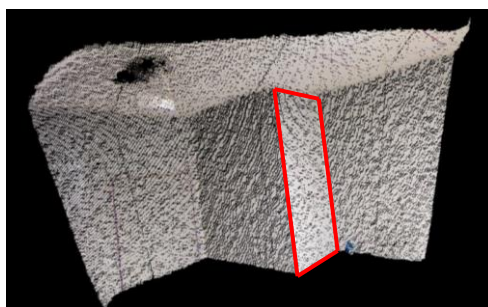


図 13. 元データ

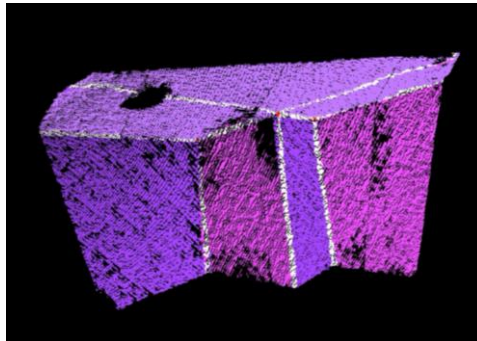


図 14. すべての面の取得が成功したデータ

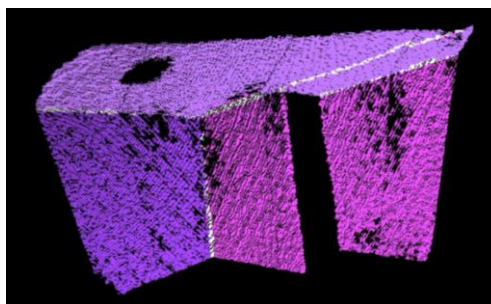


図 15. 面が欠落したデータ

点群量と速度の関係は表 3 に示してある通り点群量が少ないほど取得速度は速くなるので、面の取得にどの程度時間をかけたかでパラメータを変化させる必要がある。

4.3.2. 深度映像における平面取得精度検証

Kinect から得られる映像からリアルタイムに平面パラメータを取得することが出来るのかどうかと、その精度について検証を行う。図 16 左の空間について面の取得を行い、平面パラメータと、3 枚の面が重なる位置について考察を行う。この実験では点群量を 1/256(30 万個から 1200 個)にダウンサンプリングし、パラメータの角度を 2 度間隔で取得した。



図 16. 取得した三次元空間

上記のパラメータで 3D-Hough による平面検出を行い続けた所、ほぼ同一の平面パラメータを 2fps の速度で安定して取得することが出来た。Kinect を一点に固定して映像を撮り続けた際に、点群量の一番少ない面に関してはごく僅かにパラメータが変化し、また元のパラメータに戻るという事象が度々発生した。この事から、Kinect から取得する事の出来るデータにはフレーム毎に若干ノイズが発生し、それがパラメータの取得精度に微小ながら影響を与える事が判明した。点群量を増やした状態で同様の実験を行った際、パラメータ変動の発生回数が減少したがフレームレートが減少した。

実空間上で距離 198cm の位置に存在する 3 枚の平面がなす交点を、3D-Hough を用いた手法では距離 201cm で取得する事が出来た。この様に、誤差数センチ内で特徴点を検出する事が可能である。この様に本提案手法ではリアルタイムで 3 枚以上の面の取得、3 枚の面の交点を特徴点として取得することが可能である。

リアルタイムで面を取得するという目標のため、点群量はダウンサンプリングされ、角度間隔も広く取られている。そのため、空間内に小さい面が多く含まれている場合はその面を検知することは難しくなっている。

4.3.3. 複数枚深度画像における平面取得精度、時間検証

単数のセンサでは、一度に屋内空間の全体の平面を取得することが出来ない。そこで、複数台のセンサから深度画像を取得出来たと仮定して、複数枚の深度画像を合成した空間で 3D-Hough をした際の検証を行う。Kinect の位置情報を保持するため、Kinect を水平方向に回転させ、複数視点の点群データを合成する。取得データはカメラが水平方向に 45 度回転するごとに取得し、合計 4 枚の点群データを合成する。図 17 は取得した屋内空間を三次元描写した図である。また、この実験では 4.1.1.節で述べている仰角の限定を行った際の実験も行っている。

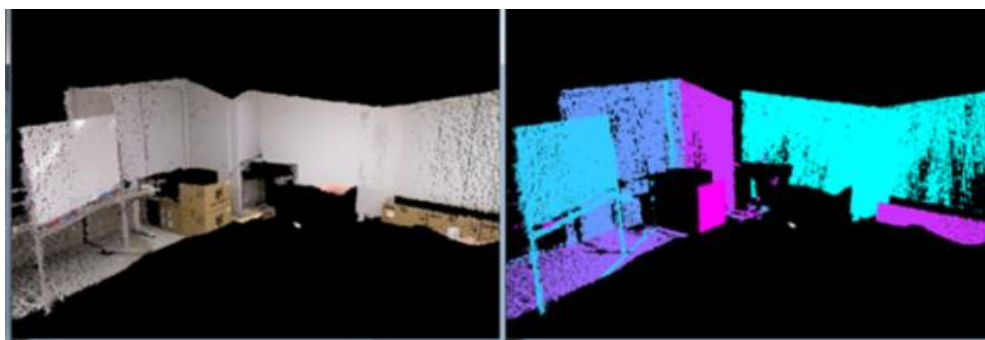


図 17. 構築した三次元空間

表 4. 単一視点, 前方位の速度実験

単位(秒)		P		
		1/16	1/64	1/256
A	1° (限定無)	157	40	11
	1° (限定有)	29	8.6	3.1

図 17 を見てみると床面が 1 枚, 壁面が 4 枚, ダンボールの側面と上面が 1 枚ずつ, ホワイトボードが 1 枚の計 6 枚の平面(壁・床)を取得することに成功した. 同じ平面上にある壁面は別個のものと認識していない. 点群量に変化しても, 同様の平面パラメータを取得することに成功した. 計測する面が取り得る仰角を限定して計測した場合と限定しない場合を比較した結果, 約 1/3 の速度で面の取得を行う事が可能である. 実験を行った空間には省略した仰角を持つ壁面が存在しなかったため, 角度を限定する前と同様に面の取得が出来た.

また別の実験として, Kinect を水平方向に 45 度ずつ回転させ, 全方位の点群情報を取得する作業を 2 箇所で行い合成した深度画像について検証を行う. 複数視点のキャリブレーションは物理的にを行い, 合計 16 枚の画像を合成する.

表 5. 複数視点, 全方位の速度実験

単位(秒)		P			
		1/4	1/16	1/64	1/256
A	1° (限定無)	2773	687	179	47
	1° (限定有)	797	199	52	17

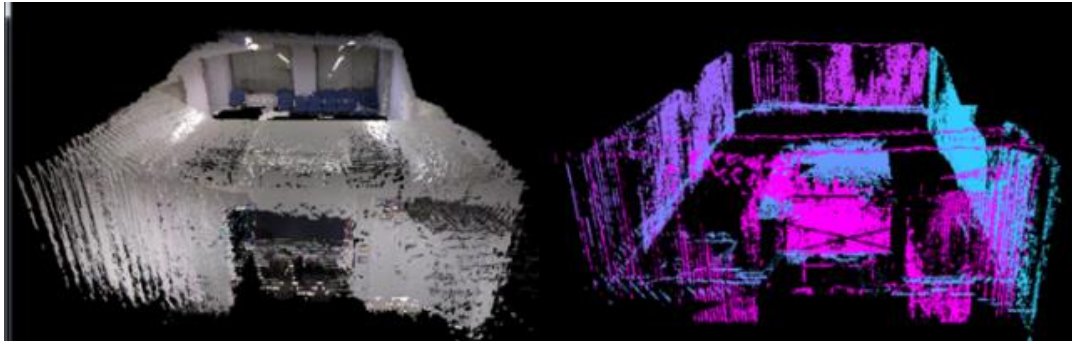


図 18. 構築した全方位三次元空間

実験を行った空間は面積が約 $6 \times 10\text{m}$ と広く，また複数枚画像の手動キャリブレーションによる実験のため，ノイズや位置情報の誤差の大きい点群が多く含まれていたが，四方の壁面と机，部屋内のテレビモニタの計 6 枚の壁面の取得を行う事が出来た．

これらの実験から，点群量さえ確保できれば複数枚の面を含む屋内空間全体や一部に関して，面の取得を行うことが可能であることが証明された．図 10, 11 を見てわかる通り，Kinect から距離が離れるほど，取得するピクセルの位置情報に誤差が生じることが判明した．これに関しては，取得した点群データにフィルターをかけることや，より精密なキャリブレーションを行う事によって改善する事が可能であると予測する．

5. ボクセルを用いた 3D-Hough による平面認識

5.1. 提案手法

Kinect から得られるデータを学習し、より精密な三次元空間のデータを形成した後に、3D-Hough による平面認識を行う。Kinect から得られたノイズの多い深度情報は 2.4.2.節で述べた符号付距離関数として、ボクセル空間に保存される。この時、既にボクセル空間がデータを保持していた場合は、過去のデータと最新のデータを照らし合わせて値の修正を行う。次にボクセル空間上に保存された符号付距離関数から、物体の表面の位置同定を行う。格子線上で符号距離関数の符号が入れ替わる位置を検索し点を打っていく。一軸に平行な格子線上の点群の取得を行った場合、その軸と平行に存在する空間内の面がなす点群が十分量取得出来ない。そこで、 x 軸、 y 軸、 z 軸の三軸上全ての軸に平行な格子線上に存在する表面を点群として検知する。

4 節で行った提案手法と比べると使用する空間データに欠落やノイズデータが少なく、点群一つ一つが小さい誤差で取得されているため、3D-Hough を行うとより細かい平面を取得することが可能になる。また、誤差の大きい点が少ないので近傍として削除しなければいけないノイズデータも軽減され、今まで近傍点として削除されていた平面の検出も可能である。

Hough 変換の特徴は、平面の検知は連続した点群に対しての検出ではなく、平面パラメータに一致する点群の総数であることである。このことによって、目視で確認出来る平面とは違う結果で平面取得してしまう可能性がある。例えば、離れた位置に同じ高さを持つ箱を置いた空間で 3D-Hough を行ったとする。二つの箱の平面は同様のパラメータを持つため、平面は一つの面として取得される。このような現象を下記の図 19 に示す。

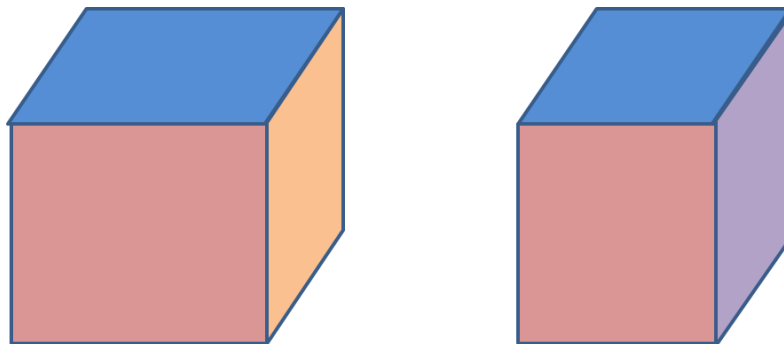


図 19. 同一のパラメータを持つ二つの面

図 19 は二つの高さを持つ箱を、距離を離して平行においた時の図である。青色の面と赤色の面は 3D-Hough を行った際には同一のパラメータを持つ点群の集まりなので、一枚の面として扱われる。このような場合に面を二つに分別して検知するため、下記のような手法を提案する。

本来、頻度グラフから平面パラメータを決定した後、近傍点の削除をするまでの手順は下記のアルゴリズムの样になっている。

```

for(index 全ボクセル空間の検索)
if(index == 未検知の表面)
  if(indexの点群は平面パラメータの近傍)
    index = 平面パラメータに一致している点群(検知済)
    for(theta 0°~360° )
      for(phi 0°~180° )
        rho = index, theta, phiを用いて求める
        counter[theta][phi][rho]ー

```

図 20. 頻度グラフから近傍の点を削除するアルゴリズム

未検知の表面とは、ボクセル空間にある点群がまだパラメータに一致していない表面である。

このアルゴリズムを、距離の離れている点群の集まりは別のグループとして検知することが出来るように改良を加える。改良を加えたアルゴリズムを下記に示す。

```

currentGroup= 0 //検索するグループ
groupNum= 0 //グループの数
for(index 全点検索)
  if(index == 未検知の表面)
    距離rho = theta_max, phi_maxを用いて求める
    if(rhoとrho_maxが近傍の距離未満)
      if(新規のグループ)
        groupNum ++
        currentGroup=groupNum
      else (既存のグループ)
        currentGroup= 現在の点
    for(index2 indexの周辺のボクセルの検索)
      if(index2 == 未検知の表面かつindex2の点群は平面パラメータの近傍)
        if(すでに検知のグループ)
          for(index3 全点検索)
            グループの合成
          else(未検知のグループ)
            index2=グループに追加
  最大頻度のグループの検索
  最大頻度のグループを面として検知
  最大頻度のグループの近傍削除

```

図 21. 平面の分割と近傍点の削除を行うアルゴリズム

上記のアルゴリズムにより、同一の平面パラメータを持つ点群を、距離を用いてグループ化し、別々の平面として検知させることが可能になる。しかし、このアルゴリズムを用いると、同一の平面でありながらも平面の途中に別の物体が存在して平面を分割してしまう場合や、データの欠落の発生により別々の平面グループとして検知してしまう現象が発生する。この事象例を図 22 にて示す。

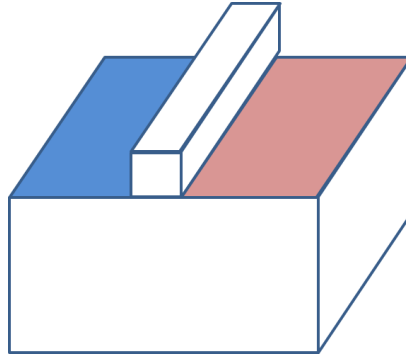


図 22. 同一のパラメータを持つ二つの面

上記の図 22 では、一つの箱の上に別の物体が置いてある状態である。箱の上部は一枚の平面であるにも関わらず、平面中心に物体が存在するせいで平面が左右に分割され、別々の平面として認識されてしまった状態である。

このアルゴリズムは、平面をグループ化する際に連続する点群であるとみなす距離によって取得することの可能な平面形状が変化する。距離が短いと同一の平面を別分割して検知してしまう可能性が発生し、距離が遠いと実際は平面ではないが実際の平面と同一平面パラメータを持つ点群を合成して一枚の面として検知してしまう問題が発生する。

このアルゴリズムについては、5.2.6.節で平面の分割アルゴリズムを実施しない場合と実施した場合の比較実験を行う。

5.2. 実験, 考察

事前に取得した表面の点群データから提案手法による平面の検出をし、その精度について検証を行う。この実験でのボクセル空間は 8mm 間隔で、約 $4\text{m} \times 4\text{m} \times 2\text{m}$ の空間としている。例として図 23 に Voxel 空間内の取得した三次元データ内の表面の位置を点群として描画した画像を示す。

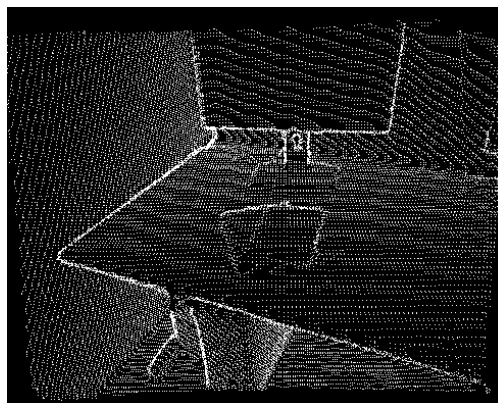


図 23. 取得した空間データ内の点群

以降、空間データや平面取得に関わるパラメータを変化させながら平面取得に関わる実験を行っていく。実験では変更されるパラメータが明記されていない場合は下記の表 6 に示している値を用いる。

表 6. 使用するパラメータ

点群間隔	1
角度間隔	1
面の距離	300
削除する近傍点	3
面とみなす近傍点	2

点群間隔とは、空間内に存在する点群のダウンサンプリング率を表す。点群量のダウンサンプリングは、ボクセル空間の点群を検索するボクセル数の割合を変化させることで面の検知の高速化を図ることが可能になる。

明記されていない場合は全ての点群に対して 3D-Hough を行っていく。角度間隔とは平面パラメータを頻度グラフに保存する際に、また平面パラメータを決定する際に用いる角度の間隔である。仰角 0~180 度、方位角 0~360 度なので、角度間隔が 1 度の時に頻度グラフは $360 \times 180 \times \text{距離}$ の入れ子を持つことになる。面の距離は、点を通る平面と原点との距離である。1cm 間隔で 0cm から 300cm までの入れ子を用意し、頻度グラフに用いる。削除する近傍点は、平面パラメータが決定した時に、その平面パラメータ付近に存在するノイズとして削除する近傍点までの距離を表す。この距離が広いと、ノイズデータを除去出来るが取得した平面パラメータ付近に存在する細かい平面データをノイズと同時に削除してしまう可能性がある。面とみなす近傍点とは、平面と平面のグループとみなす点群との距離となる平面との距離のことを指す。点群平面の距離がこの近傍点未満の点群を以降の図では描画していく。この距離を大きく取ると、平面削除される近傍点が減り、面の面積が増える。この値を大きくしすぎると、ノイズデータまで平面の一部であるとして検出してしまいう可能性が発生する。

5.2.1. 平面取得速度実験

・異なる空間データ

異なる空間データを用意し、パラメータを変化させながらその空間を 3D-Hough し、平面パラメータの取得にかかる時間について検証を行った。実験空間は下記の図 24 に示してある通りである。

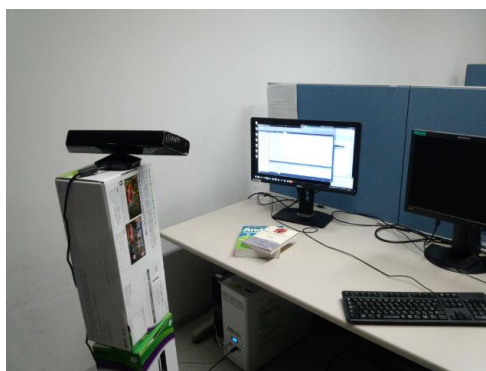


図 24. 実験空間

表 7. 平面取得にかかる時間

	data1	data2	data3
空間内の表面をなす点群	292287	57559	53541
計測時間(s)	9948	2048	1945

空間によって学習データ量が変化するため、各データ内に存在する物体の表面をなす点群量は大きく違う。Voxel 空間は $4 \times 4 \times 2\text{m}$ のスペースを確保しているが、すべてのスペースに対して表面パラメータの取得をしているとは限らず、センサで取得していない位置に関しては空間として扱っている。また空間内の物体の形が複雑であることで表面が多いと点群量が増加し平面取得にかかる時間が長くなり、逆にデータ内に空間や体積の大きい物体が多いと点群量が減少し平面取得にかかる時間が短くなる。

・異なるパラメータ

次に、一つの空間データに対して取得する平面のパラメータの角度間隔を変化させながら、平面の取得にかかる計測時間について検証を行った。この空間には表面をなす点群が 53531 個検出された。

今回角度間隔は 2 度, 1 度, 0.5 度と設定した。実験結果より、平面パラメータ取得にかかる時間について下記の表 8 に示す。

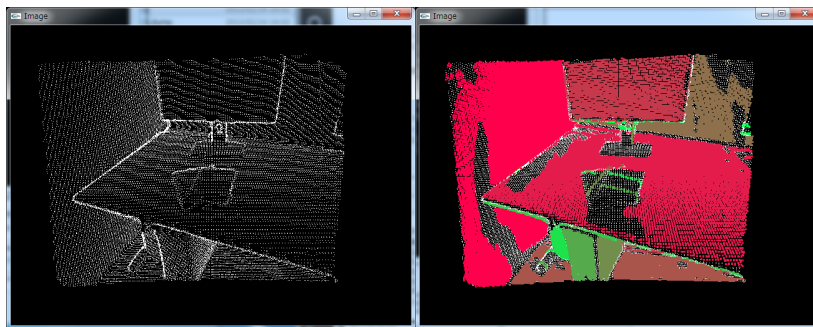


図 25. 取得できた平面

表 8. 平面取得にかかる時間

	Angle = 2	Angle = 1	Angle = 0.5
計測時間(s)	284	1617	6334

角度間隔は 3D-Hough を行う際に使用する頻度グラフのサイズに関係する。角度間隔が 1 度の時は 360×180 回点群を通る面への距離を計算するのに対して、角度間隔が 0.5 度の時は 720×360 回計算が必要になる。この空間データでの実験では、1 度間隔でパラメータを取得すると 5 分弱で計測が終了するのに対して、0.5 度では約 27 分、0.25 度間隔では 1 時間 45 分もの計測時間が必要になる。計算量が増加するにつれ、平面取得にかかる時間も増加する。

実験結果から、空間内の表面をなす点群量は多ければ多いほど、角度は細かければ細かいほどより細かい面の検知や高精度のパラメータで平面検知を行う事が可能である。また、空間内に存在する平面の数が少なく、ノイズが少ないデータの方がパラメータの決定、近傍点の削除を行う際の計算量が減る。

次に，空間内の表面となる点群量をダウンサンプリングして平面取得に要する時間について実験を行う．この実験で，点群量が変わるのでパラメータとして検知される最低頻度を点群量に合わせて適切な値に変換している．実験結果より，平面パラメータ取得にかかる時間について下記の表 9 に示す．

表 9. 平面取得にかかる時間

	P = 1	P = 1/2	P = 1/4	P = 1/8
空間内の表面をなす点群	53541	26678	13283	6648
計測時間(s)	1945	978	492	193

実験結果より，点群量が減少させると，それに応じて計算量が減少するため平面取得にかかる時間も減少することがわかる．いずれの実験結果から，4 節同様に秒単位での三次元空間内の平面の抽出を行う事は難しいと考えられる．

5.2.2. 多数平面取得精度実験

ある空間データに対して，空間内に存在する面をどの程度の精度で取得することが可能か検証を行った．実験は再現性を検証するために数種類の空間データに対して同様の実験を行っている．下記の図 26 は実験に使用した空間と，作成された空間データ，取得した平面を示している．

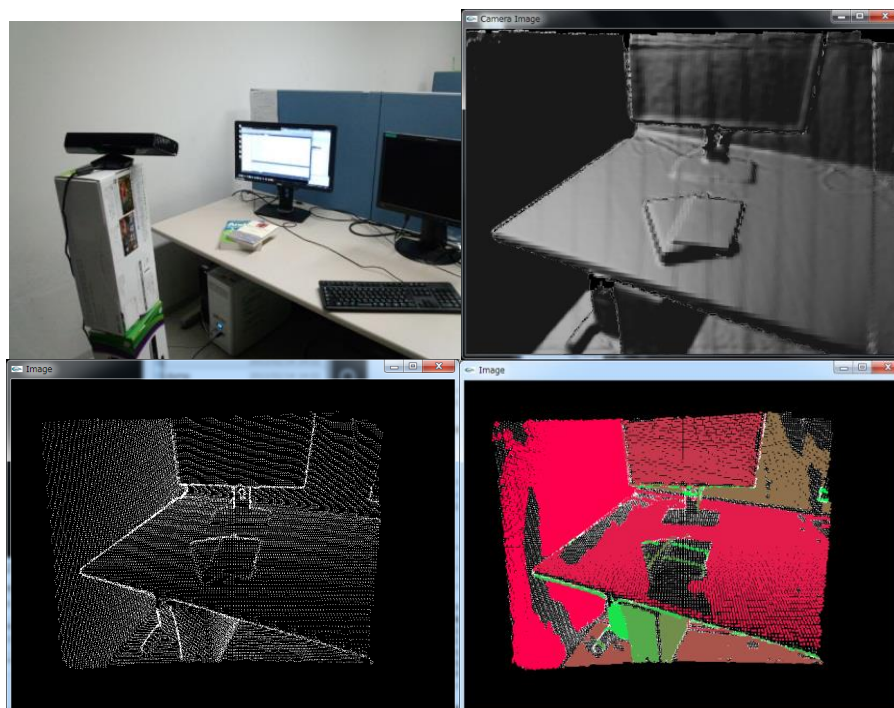


図 26. 実験に使用した空間(左上)と実験データ(右上)
実験データを点群にしたもの(左下)取得した平面(右下)

図 26 の実験では以下の平面を取得した.

- ・机
- ・壁
- ・床
- ・モニター
- ・パーティション(奥の壁)
- ・床下の PC の側面
- ・床下の PC の前面

空間内に存在する大きな平面形状(机, 壁, 床等)は概ね正確に取得することが出来た. また, モニターの画面や PC 本体の側面と前面等も同時に取得することが出来た. 図内の卓上に置いた本については机の近傍として削除されてしまっている.

次に別の空間データを用いて同様に実験する, 下記の図 27 は実験に使用した空間と, 作成された空間データ, 取得した平面を示している.

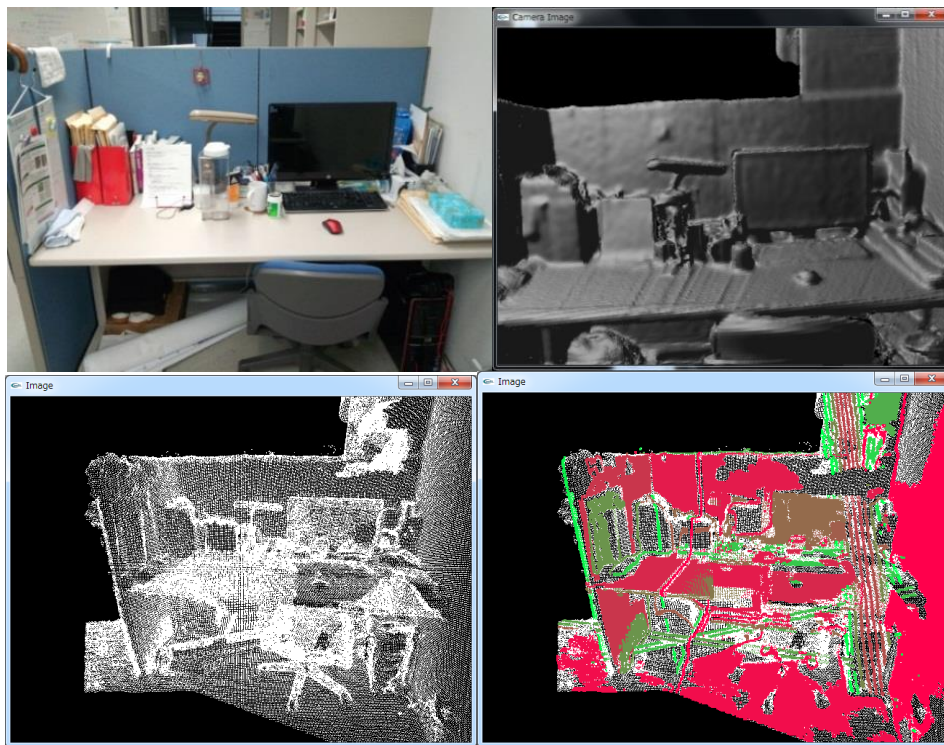


図 27. 実験に使用した空間(左上)と実験データ(右上)
実験データを点群にしたもの(左下)取得した平面(右下)

図 23 の実験では以下の平面を取得することが出来た.

- ・ 壁面(図内右)
- ・ 床面(図内下)
- ・ パーテーション(正面)
- ・ 机
- ・ パーテーション(左)
- ・ ノイズ
- ・ ノイズ
- ・ モニタ
- ・ モニタの裏側のパーテーション
- ・ 机の下の方
- ・ ノイズ
- ・ ホワイトボード(図内左のパーテーションの位置にあるもの)
- ・ ノイズ
- ・ パーテーションの奥の棚
- ・ 床に置いてある紙筒の表面の一部
- ・ ノイズ

取得した平面は計測した点群の頻度の高い順に記載している. 今回は, 平面を持たない小物(画面中央の電気スタンド等)や, 小さい面積の平面(机の右端にある書類, 小箱等)等, 3D-Hough で誤検知の起こしやすいと思われる空間データを用いて実験を行った. 空間内に存在する大きな面の取得に成功したが, 最大頻度の点群量が少なくなるにつれ, 空間内に検知されずに残っている平面よりも空間内の曲面が持つ点群のパラメータの頻度数が上回ってしまい, 平面の誤検出を行ってしまう事象が発生した. 発生した誤検知の例を下記の図 28 に示す.

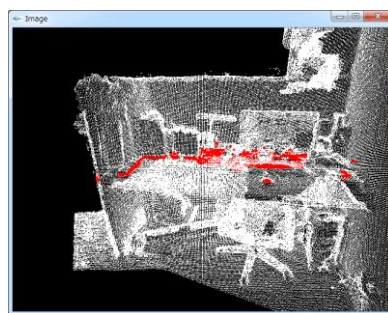


図 28. 誤検知したパラメータに一致する点群

以上の実験から, 本提案手法を用いて空間内にある 10 枚以上の面のパラメータを取得することを可能とした. 実験で使ったパラメータは基本的に表 6 で述べたものであるが, このパラメータを変化させる事によって取得可能な平面パラメータが変化することが予測される. そこで以降の節では表 6 に記載したパラメータを変化させることで取得出来る平面パラメータがどの様に変化するかどうか実験を行う.

5.2.3. 角度間隔変化による平面取得精度実験

ある空間データに対して、パラメータを取得する角度の間隔を変化させながら取得できる平面パラメータの変化について検証を行った。下記の図 29 に実験に使用した点群データと取得した平面を、表 10 に取得した平面パラメータを示す。

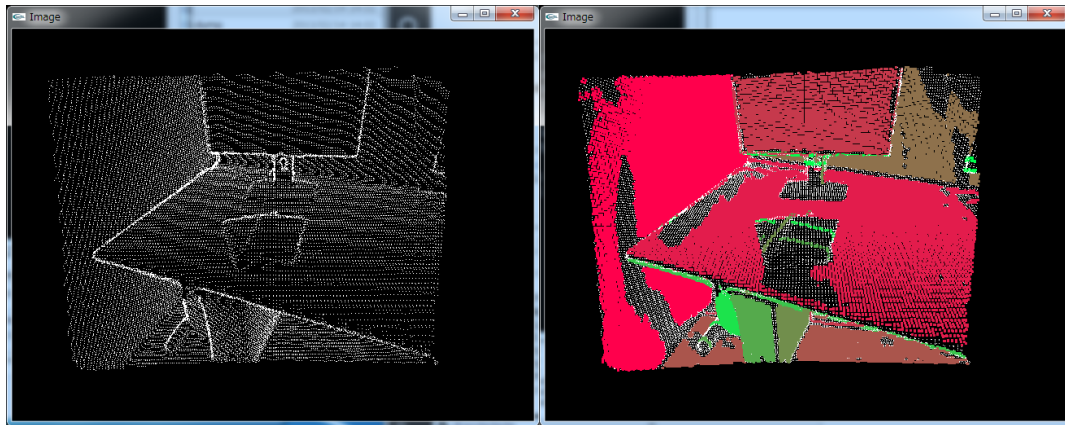


図 29. 実験データ(左)と取得した平面

表 10. 取得した平面パラメータ

	plane	0	1	2	3	4	5	6	7	8
4 度	theta	92	168	72	92	56	168	52	72	56
	phi	120	72	32	120	40	72	44	72	44
	rho	54	90	110	120	127	38	60	6	69
	count	6926	5477	2697	2090	2270	1558	1275	998	521
2 度	theta	92	170	72	92	54	170	52	74	56
	phi	122	74	32	122	40	72	42	78	44
	rho	57	87	110	123	127	39	64	2	69
	count	11400	7493	2582	3002	2429	1610	1430	938	712
1 度	theta	91	169	75	92	55	167	51	75	54
	phi	122	73	33	123	41	72	42	78	43
	rho	57	89	110	125	126	37	64	2	71
	count	11400	11761	3160	2811	2508	1746	1476	987	728

パラメータを取得する角度の間隔を変化させた場合、取得出来る平面パラメータが大きく変化する事象は起こらなかった。角度の間隔が狭ければ狭いほど、パラメータに一致する空間内の点群量は増えた。角度間隔を大きくした際には精密な角度パラメータでは取得出来ていた平面一部、特にデータに歪みが発生しやすい四隅の点群がパラメータに含まれる点群として検知することが出来なかった。またこの事象は平面が大きくなるにつれて発生しやすくなっている。この事より、角度間隔を広く取った際には、より高速に空間内の平面のパラメータを取得できるが、実際には平面をなす点群を見逃してしまい、結果的に正確な平面形状を取得することが難しくなる可能性がある。下記の図 30 は、4 度間隔で取得した面と 1 度間隔で面を取得した際の一致した点群量の差を示している。

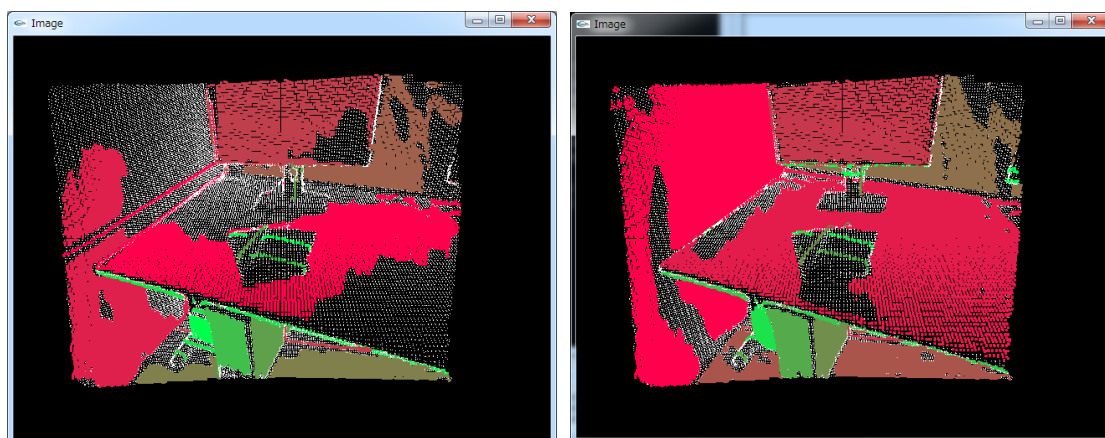


図 30. 4 度間隔で面を取得した時のパラメータ(左)と
1 度間隔で取得したパラメータ(右)

図 30 の通り、4 度間隔で取得した平面パラメータでは、机や壁をなす平面パラメータに一致する点群量が 1 度間隔で取得した平面パラメータとは大きく変化することが図から見て取れる。今回の実験では平面として検知されなかった点群は近傍点として削除されたが、今回の実験以上に大きい面を取得した際などに今以上の誤差が出た場合に、削除する近傍点から点群がはみ出し、はみ出した点群で新しい平面パラメータを持つてしまうことが推測できる。

実験結果より、大まかな平面の位置と枚数を取得する際には平面パラメータの間隔を広げることが可能であるが、正確な平面形状を取得するためには、角度間隔は 1 度未満で行う必要があることが判明した。

5.2.4. 点群量変化による平面取得精度実験

ある空間データに対して、空間内のパラメータをダウンサンプリングし、どの様に平面取得精度が変化するかどうか実験を行う。今回は点群量 1/1, 1/2, 1/4, 1/8 で実験を行った。

表 11. 取得した平面パラメータ

1/1			1/2			1/4			1/8		
theta	phi	rho	theta	phi	rho	theta	phi	rho	theta	phi	rho
169	73	89	169	73	89	169	73	89	169	73	89
91	122	57	91	122	57	91	122	57	91	122	57
75	33	110	75	33	110	75	33	110	75	33	110
92	123	125	92	123	125	92	123	125	92	123	125
55	41	126	54	40	127	55	41	126	55	41	126
167	72	37	167	72	37	167	72	37	52	42	64
51	42	64	51	42	64	51	42	64	167	72	37
75	78	2	75	80	5	73	74	3	75	80	5
54	43	71	56	41	76	89	101	1	57	44	69
88	101	1	88	101	1	87	123	53			
			131	100	11	58	41	76			
			88	123	53						
			178	89	26						

空間内に存在する平面は以下の通りである.

- ・机
- ・壁
- ・床
- ・モニター
- ・パーティション(奥の壁)
- ・床下の PC の側面
- ・床下の PC の正面
- ・本 A
- ・本 B

いずれの点群量で実験を行っても、大まかな平面のパラメータは取得することが出来た. また取得した平面パラメータを見ていくと、4 枚目までのパラメータは完全に一致するパラメータを取得しているのに対して、大きい面が無くなって小さい面を取得する際、すなわち頻度が十分に取得できない空間になった時には、ダウンサンプリングした時のパラメータは全点群を用いた際のパラメータとは変化していることが分かる. 実験より、空間内の大きな面のみを取得する際には、点群量を 1/8 まで下げることが可能であるが、小さい面の検知には誤差が生じやすくなってしまうことが判明した.

5.2.5. 近傍点変化による平面取得精度実験

ある空間データに対して、取得したパラメータの付近に存在するノイズデータを削除する近傍点を変化させながら、どの様に平面取得精度が変化するか実験を行う。空間内に平面を持つ小物を置いて、平面形状の取得検証を行う。空間内には床、壁、本 A、本 B、箱が存在し、実験に利用した本の厚さは、本 A は 1.3cm、本 B は 3.6cm で、箱は 28*21*9cm である。本は積み上げられており、床面の上に 3.6cm の本 B、1.3cm の本 A の順に配置している。実験環境を図 31 に示す。

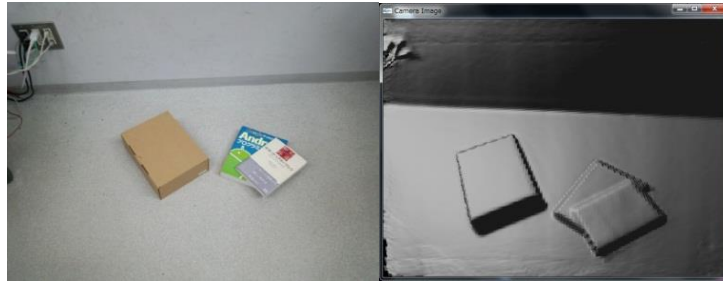


図 31. 実験環境と実験データ

実験はパラメータの検知後、削除する近傍点の範囲を変化させて行った。変化させる範囲は 1cm 未満、3cm 未満、5cm 未満とする。

表 12. 取得した平面パラメータ

	近傍	1cm 未満	3cm 未満	5cm 未満
床	theta	93	93	93
	phi	127	127	127
	rho	85	85	85
	頻度	16259	15674	15035
本 B	theta	95	95	
	phi	123	129	
	rho	77	83	
	頻度	865	991	
本 A	theta	95		
	phi	123		
	rho	76		
	頻度	1487		
箱	theta	90	90	93
	phi	126	126	127
	rho	76	76	76
	頻度	1487	1467	1415

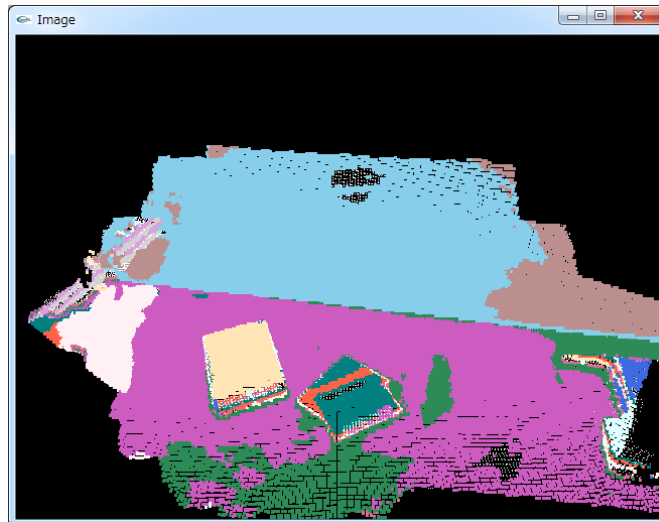


図 32. 近傍 1cm 未満の時の平面

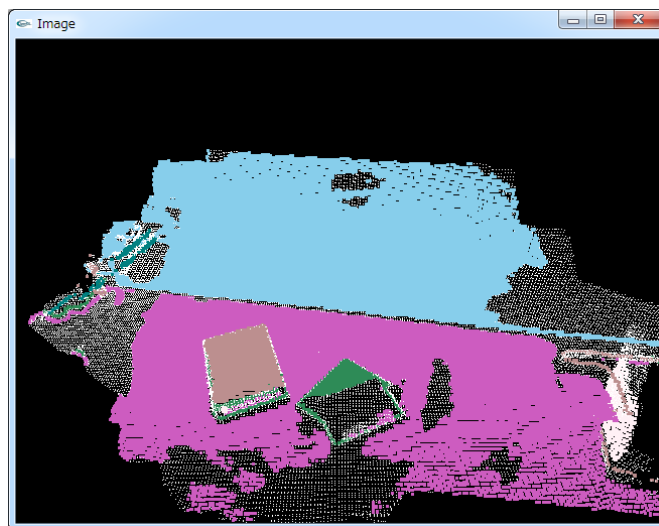


図 33. 近傍 3cm 未満の時の平面

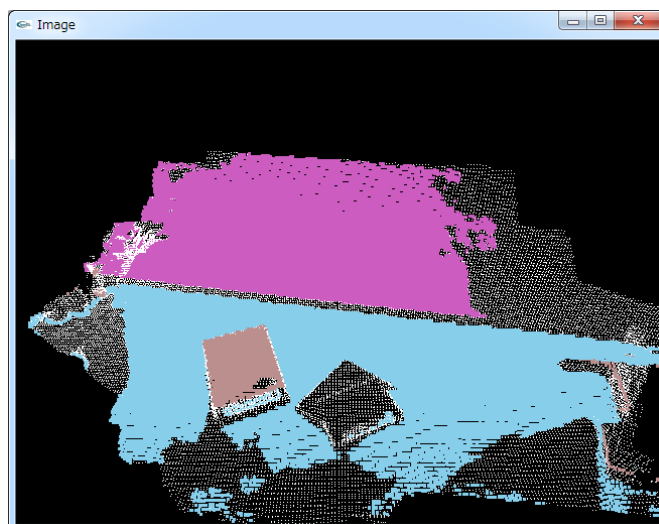


図 34. 近傍 5cm 未満の時の平面

表 12 ではそれぞれの空間で取得することの出来た平面のパラメータを表示している．パラメータの取得が出来なかった場合は斜線を引いている．

平面と削除する点の距離が 3cm 未満の時と 5cm 未満の時に空間内の平面を正確に検知することが出来なかった．近傍の距離が 3cm の時には，床面をパラメータとして決定した後，次点で平面をなす点群の数が多かった本 A を検知し，その近傍として本 B を形成する点群を削除してしまったと推測できる．また，近傍の距離が 5cm の時には，本 A と本 B の厚さを合わせて 4.9cm であったために，床面の検知を行った後に，その近傍のノイズとして削除されてしまったと推測される．

平面と削除する点の距離が 1cm 未満の時に，用意した物体全ての平面の検出を行う事が出来た，だがしかし，空間内に存在する平面の枚数が 5 枚(壁，床，箱，本，本)あるのに対して，12 枚の平面を取得してしまった．取得した全ての平面パラメータについて下記の表 13 に示す．

表 13. 近傍 1cm 未満の時に取得した平面パラメータ

plane	theta	phi	rho	count	検知した面
0	71	42	88	17217	壁
1	93	127	85	16259	床
2	71	42	89	4730	壁(ノイズデータ)
3	93	127	84	4519	床(ノイズデータ)
4	97	128	83	3176	床(ノイズデータ)
5	95	123	76	1490	本 B
6	90	126	76	1487	箱
7	95	123	77	865	本 A
8	130	114	1	775	ノイズ
9	174	74	44	755	ノイズ
10	93	127	83	666	本 AB の側面
11	131	115	1	621	ノイズ
12	177	75	52	594	ノイズ

削除する近傍点までの距離を減らした結果，今までノイズデータと共に削除されていた平面形状を取得することが可能になったが，空間内のノイズデータを削除しきれず平面として誤検知してしまう結果となった．

結果より，本提案手法により平面を取得する際は近傍 3cm 付近までを削除することで，ノイズデータを除去しつつ精密な平面のパラメータの取得をすることが可能であることが判明した．しかし，Voxel 空間を用いた空間データの精密さは空間データ取得の際に要した学習量に依存し，空間データを取得する時点でずれや誤検知をしてしまった空間に今以上に削除する近傍点の範囲を広げる必要がある．これは空間データ内に存在するノイズデータや空間の歪みを事前に削除することで解決可能であると推測できる．また，空間デー

タの精密さが向上することで、近傍3cm前後では取得することが不可能であった厚さ1.3cmの本がなす平面の検出を可能にすると推測できる。

5.2.6. 平面パラメータ分割アルゴリズムを用いた複数枚平面取得

大小様々な平面のある空間について平面パラメータ取得の実験を行った。5.1節で述べた平面分割アルゴリズムを利用した場合と利用しない場合の2種類実験を行う。本実験では削除する近傍点範囲を2cm未満に設定している。

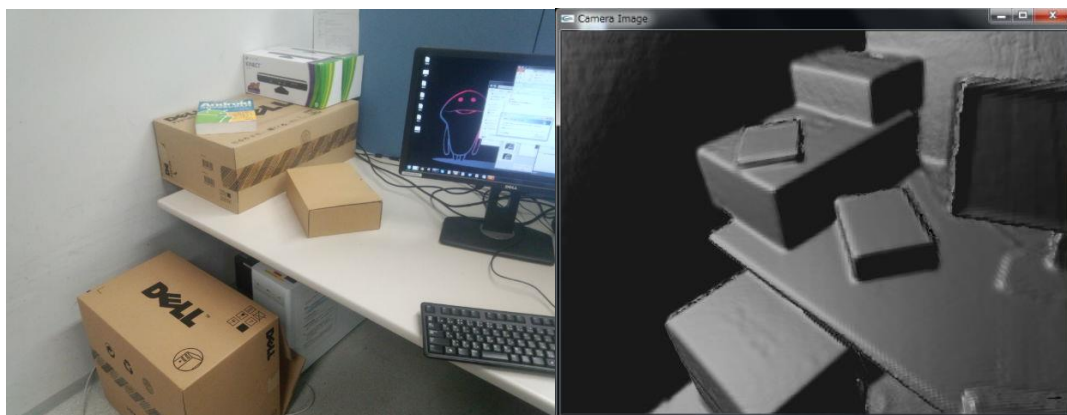


図 35. 実験環境(左)と空間データ(右)

空間内には机や壁などがなす大きな平面と、小箱や本がなす小さな平面等 15 枚以上の様々な面を用意した。

まず、平面パラメータを距離で連続する点群ごとに分割しない際に取得出来た平面について下記の図 36 に示す。図 36 で示している面 6 や面 12 の様に、空間内の複数の平面を一枚の面として検知してしまっている。原因は 3D-Hough 変換によって取得することの出来る平面パラメータは連続している点群の検出ではなく、点群の総量で決まっているからである。

次に、平面パラメータを距離で連続する点群ごとに分割した際に取得出来た平面について下記の図 37 に示す。

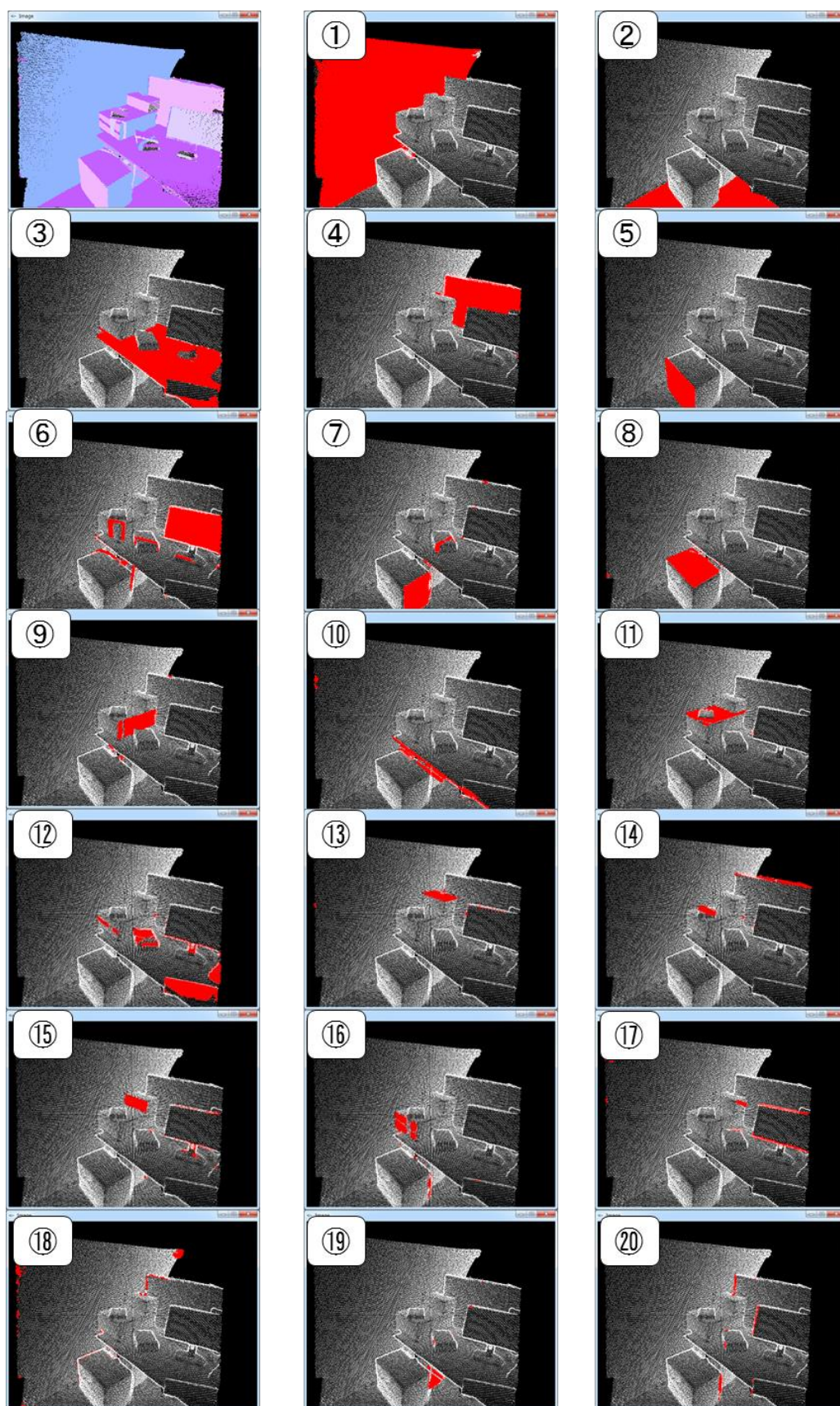


図 36. 空間内の平面

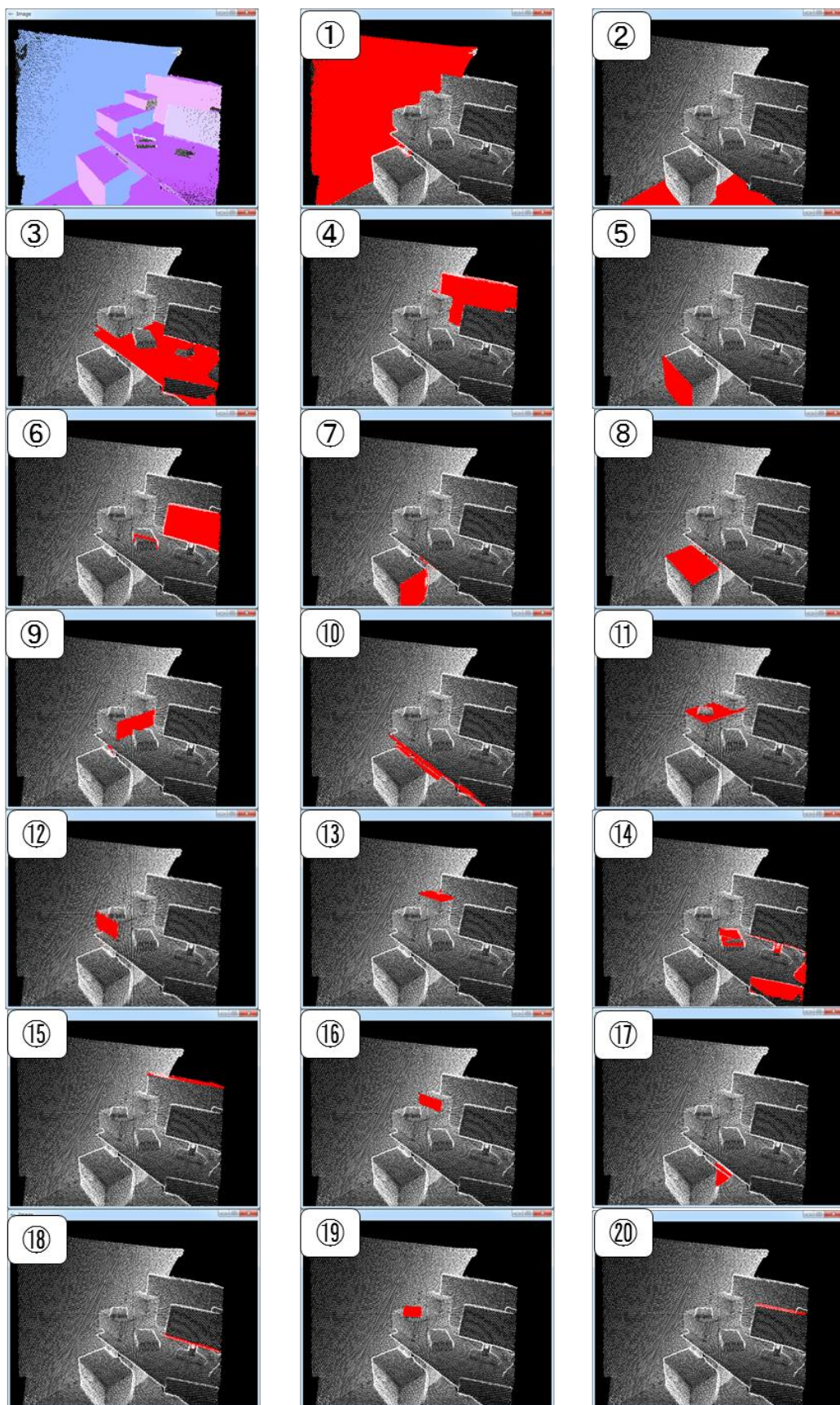


図 37. 空間内の平面(グループ化アルゴリズム:近傍 8cm)

パラメータのグループ化する距離は、ボクセル 10 個分、8cm に設定した。前述した図 36 などで見られた別平面の誤検知が、図 37 の面 6 のように解消されていることがわかる。しかし、図 37 の面 12 の様に、別の平面であっても、平面と平面の距離が近い(この場合は 8cm 未満)であった場合は未だに同一の面として検知してしまっている、そこで、グループ化する距離を更に 4cm 縮めて同様の実験を行った。下記の図 38 は取得した平面をすべて描画した図、図 39、図 40 は取得した平面データを頻度順に並べた時の図である。

取得した平面の元になる物体について、下記の表 14 に示す。

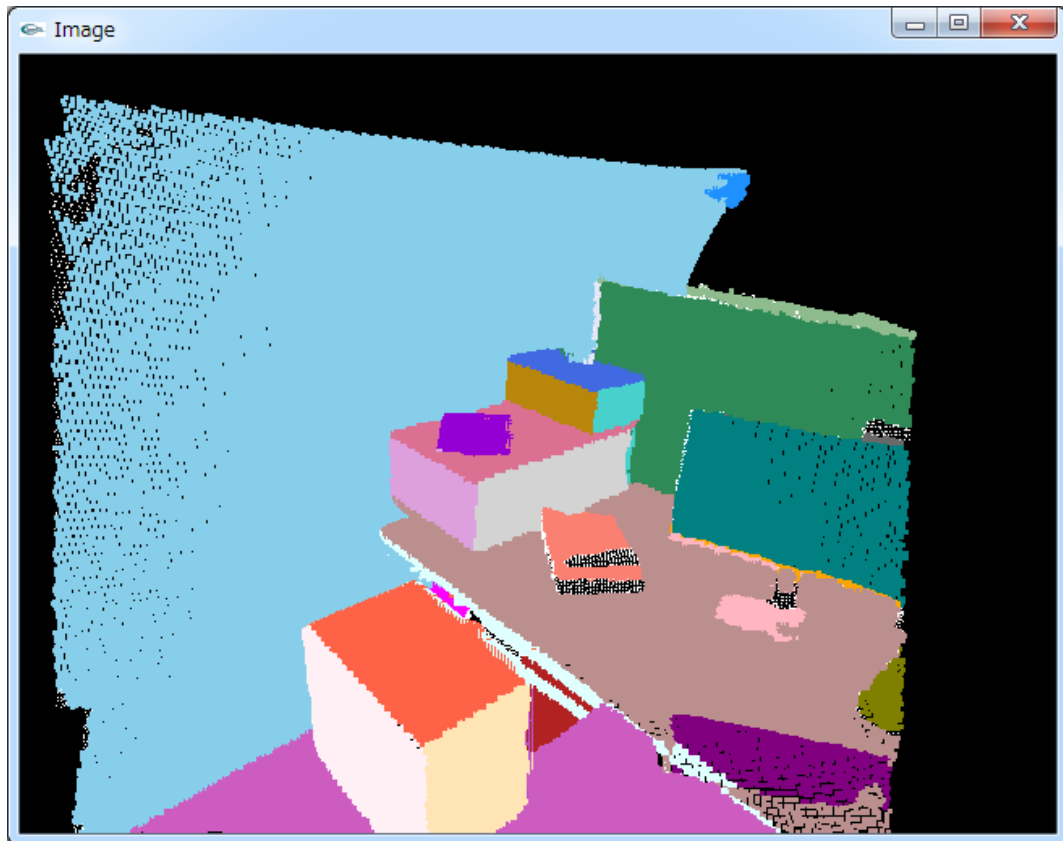


図 38. 空間内の平面の集合

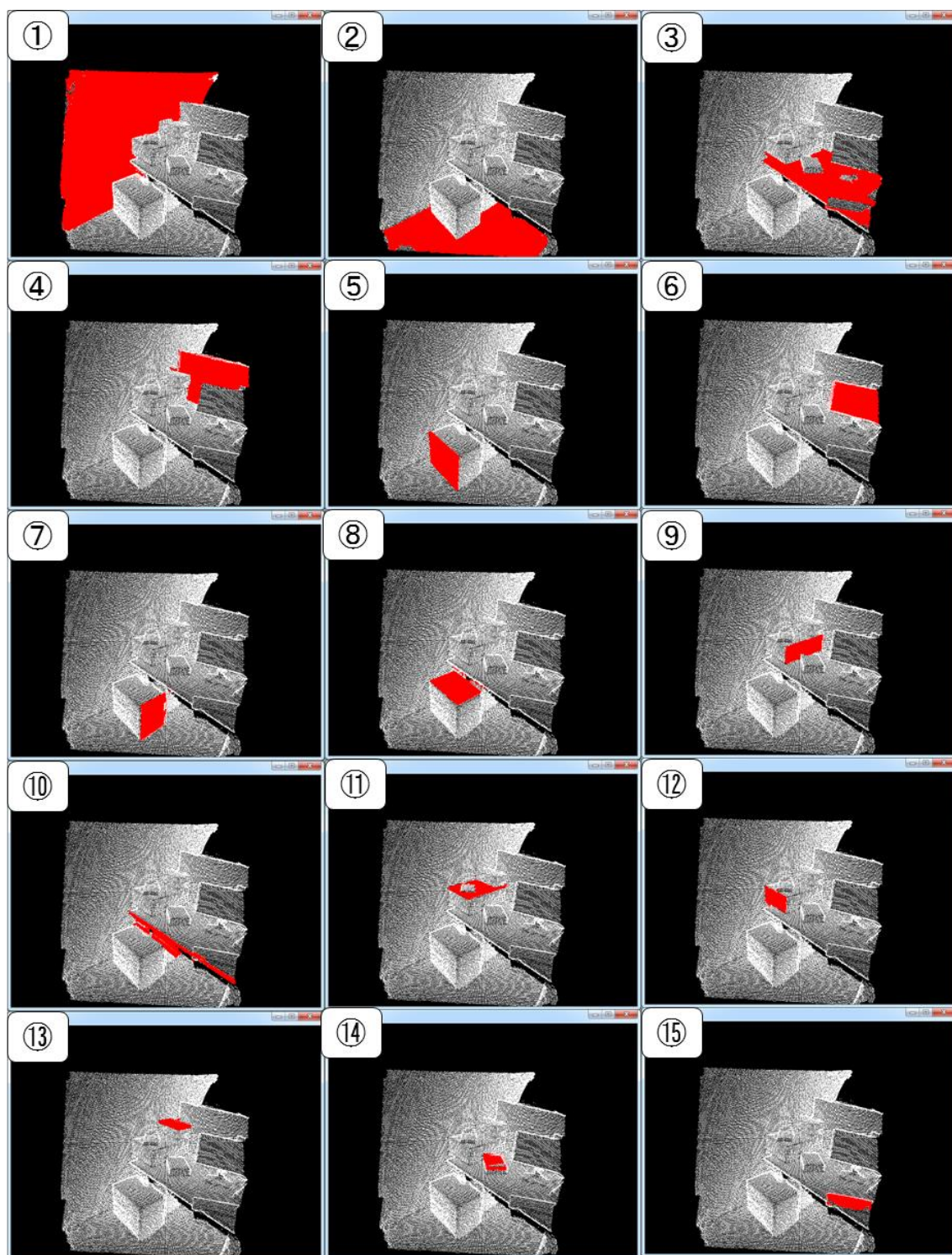


図 39. 空間内の平面(グループ化アルゴリズム:近傍 4cm)①

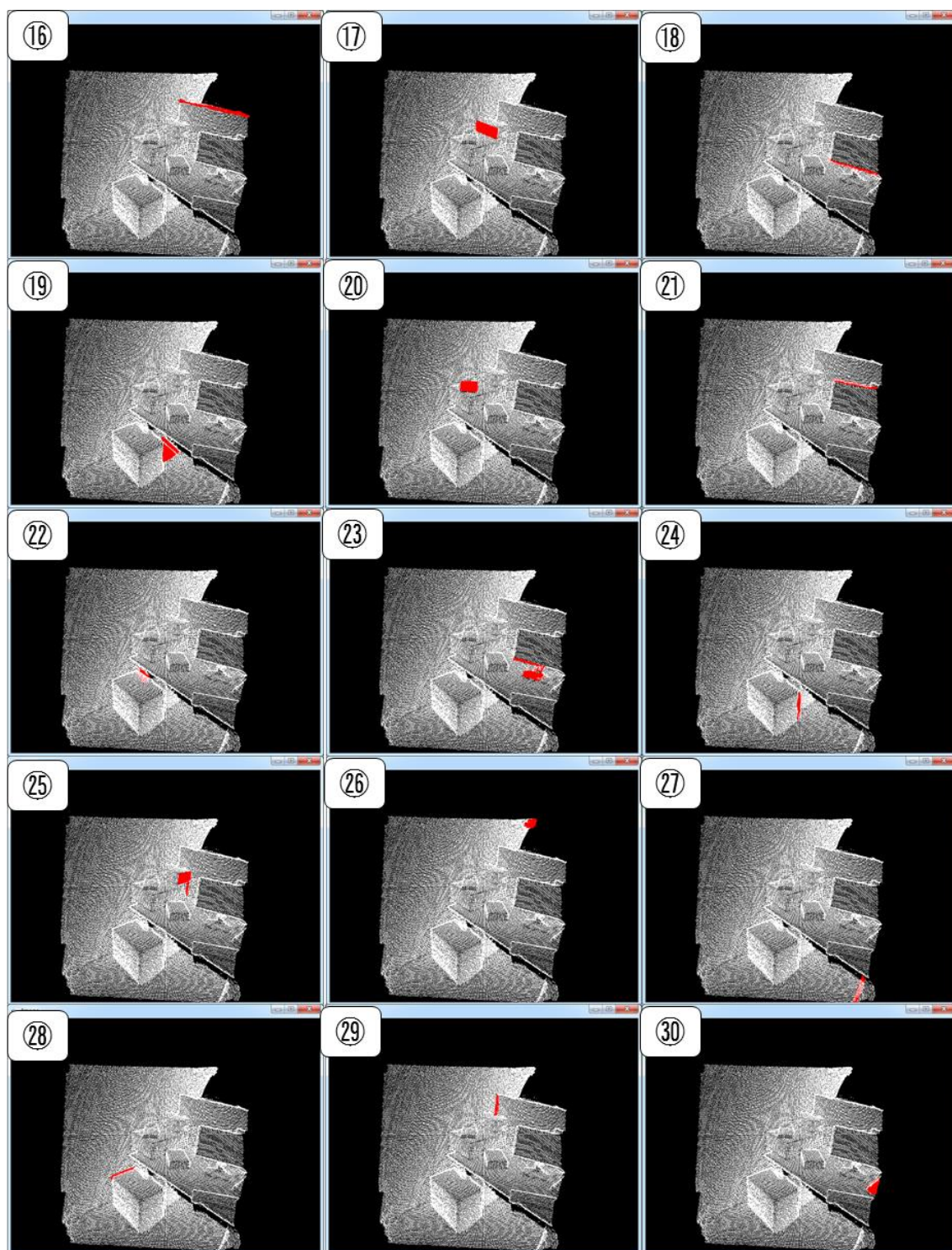


図 40. 空間内の平面(グループ化アルゴリズム:近傍 4cm)②

表 14. 取得した平面と結び付けられる物体

面 01	壁
面 02	床
面 03	机
面 04	パーテーション(奥壁)
面 05	ダンボール大(前面)
面 06	モニター
面 07	ダンボール大(側面)
面 08	ダンボール大(上面)
面 09	ダンボール中(側面)
面 10	机の側面
面 11	ダンボール中(上面)
面 12	ダンボール中(前面)
面 13	箱 A(上面)
面 14	箱 B(上面)
面 15	キーボード
面 16	パーテーション(上面)
面 17	箱 A(前面)
面 18	モニター(下面)
面 19	PC A(側面)
面 20	本
面 21	モニター(上面)
面 22	PC B(前面)
面 23	モニター台 A
面 24	ダンボール大(後面)
面 25	箱 A(側面)
面 26	壁(ノイズデータ)
面 27	パーテーション(手前)
面 28	ダンボール大(側面)
面 29	パーテーション(側面)
面 30	モニター台 B

表 14 から分かる様に、計 30 枚弱の平面形状の検出に成功した。30 枚の面のうち、面 26 の壁面の一部の四隅で歪んでいる部分を面と誤検知した以外には、概ね平面のある部分を正確に検知している。しかし、面 10、面 18、面 21、面 23、面 28、面 29 は、ノイズデータと平面データを合わせて一枚の平面として検知してしまっている。このノイズデータとは、空間データを作成する際に物体と物体の切れ間に発生した存在しない表面データである。特に面 10 に関しては、机の側面をなす点群と、PC をなす表面の一部をノイズデータで繋ぐ事で一枚の変則的な形状の面として検知してしまっている。面 10 を別角度から描画した画像を下記の図 41 に示す。

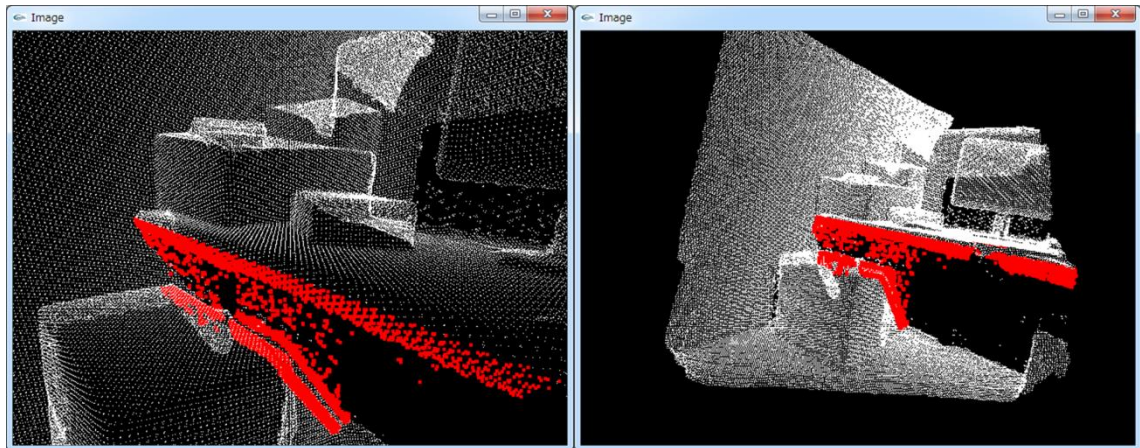


図 41. ノイズデータを繋ぎにして一つの平面として誤検知してしまったデータ

実験結果より、三次元空間内の大小様々な平面を精密に取得することを可能にした。空間内の 30 枚弱の平面を、形状も含めて正確に取得した。しかし、環境によってはノイズデータの集合を平面として誤検知してしまう事象が発生した。解決法としては、疎な密集度の平面データを取得した場合は平面が実際に存在するかどうか検証する方法などを検討する必要がある。

6. 考察

本論文では三次元空間内の複数枚の面の検出、面と面がなす交線、交点である特徴点の抽出を行った。2.1.2 節で述べた先行研究[2]で使用された航空機搭載のレーザスキャナを用いた平面検出法の相違点は下記の通りである。

- ・安価なセンサである **Kinect** を用いた平面検出を行った。
- ・屋内空間での平面形状の検出を行った。

先行研究のレーザスキャナに比べて安価なセンサである **Kinect** には、データのノイズや除法の欠落、カメラセンサの歪みなどが存在する。そのためデータの取得について手法を提案し、屋根面と比べて複雑な形状、平面パラメータをもつ平面の検出を実装した。またこの先行研究では述べられていないリアルタイムでの平面形状取得についても本論文では述べている。

また、2.2.節で述べた先行研究[3]では RGB データと深度データの両方を用いて、建物の廊下など周回できる空間の 3D マッピングを実装したのに対して、本研究では一つの部屋に対してより細かい平面形状の検出を深度データのみを用いて取得する手法について実装した。取得した点群データを蓄積、修正を行いながらする **Voxel** 空間と符号付距離関数を用いた三次元データの生成を行い、その空間内で検出した全ての点群に対して平面パラメータ取得をするための計算を行っている。

平面形状の検出に RGB データを用いた場合、平面は同色の色情報を保持している必要がある。一枚の平面が途中で色が変わってしまう場合、例えば、光の差し込みによる陰影による色情報の変化や、平面自体の色情報が途中で変化する場合には同一の平面を別個の平面として認識する。また、色情報が取得出来ない環境(暗闇等)では平面を取得できない。それに対して本手法では色情報を必要とした平面取得法なので、同じ平面パラメータを持つ点群であれば色情報が一致しなくても一枚の平面として検出する。この様に他の手法では検出出来ない平面形状を検出することが可能であるため、本手法は有用であると言える。

本論文では 3 つの平面取得法について提案、実装を行い、精度検証などを行った。

・2D-Hough による平面認識

Kinect から取得したデータ 1 フレームに対して、三次元空間を複数の二次元空間に変換するため輪切りを行い、各二次元空間で 2D-Hough を行い、平面を形成する直線の検出を行った。空間内の二枚の平面と、面と面がなす交線の検出に成功したが、**Kinect** から取得出来るデータにノイズが多く含まれていることと、輪切りを行ったことにより 2D-Hough を行う際に使用できる空間内の平面データが減少してしまったことから、空間内の複雑な平面形状、例えば三枚以上の平面、小さい平面等を検出することは出来なかった。

・素データを用いた 3D-Hough

Kinect から取得したデータ 1 フレームに対して、3D-Hough を行い平面形状の検出を行った。空間内の 5 枚以上の平面を取得することに成功した。空間データをダウンサンプリングすることでリアルタイムでの平面検出を実現した。

・Voxel 空間を用いた 3D-Hough

Kinect から取得したデータを複数合成し、深度情報の更新と修正を行う事で広範囲、高精度の 3D-Hough 空間を作り出し、その中で 3D-Hough を用いて平面形状の検出を行った。本の表紙などが形成する平面など、詳細なものに関してもより精密に取得することを実現した。また、同一の平面パラメータを持つ別々の平面をグループ化による分割を行うアルゴリズムを用いて、空間内の 30 枚弱の平面形状を検出することを実現した。

以上の実験を通して、本論文の提案する平面形状の検出は正しく機能することが示された。また、空間内で様々なサービスが提供できる基礎の構築を実現した。

7. おわりに

本稿ではライフログの計測に必要な環境の構築を行うため、Kinect を用いて取得した三次元情報付き画像から平面形状の抽出法の提案と実装を行った。平面の抽出法としては、Kinect から得られる素データに対して 2D-Hough を用いた手法、素データに対して 3D-Hough を用いた手法、3D-Hough とボクセル空間を組み合わせた手法を挙げた。CPU 環境では、リアルタイムで取得することを優先した際には 4 章の 3D-Hough による平面認識、より精密な面データを取得する場合には 5 章のボクセルを用いた 3D-Hough による平面認識が適切であると推測できる。

謝辞

研究と修士論文の作成に際して、様々なご指導を頂きました藤田悟教授と、中間発表でご指導を頂きました花泉弘教授や伊藤克亘教授を始め、情報科学部の教授の方々に心よりお礼を申し上げます。そして学会発表の際に御教授頂いた東京大学生産技術研究所の岩井将行様、多くのご指導を下さいました藤田研究室の同期、後輩の皆様に感謝の意を表します。

参考文献

- [1] Dorit Borrmann et al. , “The 3D Hough Transform for Plane Detection in Point Clouds: A Review and a new Accumulator Design”, 3D Research, Volume 2, Issue 2, pp.1-13(2011).
- [2] George Vosselman and Sander Dijkman, “3D BUILDING MODEL RECONSTRUCTION FROM POINT CLOUDS AND GROUND PLANS”, Int. Arch. of Photogrammetry and Remote Sensing, pp.37-43(2001).
- [3] Shahram Izadi et al. , “KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera” UIST '11, pp559-568(2011).
- [4] Newcombe, Richard A. “KinectFusion: Real-time dense surface mapping and tracking” , ISMAR, pp127-136(2011).
- [5] 安斎達也, 藤田悟, ”Kinect を用いた屋内空間の三次元形状の検出”, 情報処理学会 第 74 回全国大会, 1ZD-2(2012).
- [6] 安斎達也, 藤田悟, ”3D-Hough 法を用いた屋内空間の三次元平面形状の検出”, 第 11 回情報科学学術フォーラム, K010(2012).