2015 年度(平成 27 年度)

ECPⅢ
Final Report

家具シミュレーションシステムの開発

チーフアドバイザー　：　金丸　隆志　准教授

サブアドバイザー　：　矢崎　敬人　准教授

チームメンバー

G1-12025　片山 樹広

G1-12026　金子 光一

G1-12042　鈴木 徳人

G1-12044　須多　 潮

G1-12060　橋本 将吾

内容

第 1 章 緒言 （須多担当）.....1
1.1 研究背景.....1
1.1.1 研究テーマ.....1
1.1.2 3D 技術について.....2
1.1.3 レーザを使用した 3D スキャン実験.....4
1.1.4 カメラを 2 台使用した 3D スキャン実験.....8
1.2 研究概要.....12
1.2.1 製作物の決定.....12
1.2.2 家具シミュレーションについて.....15
1.2.3 ステレオ法について.....18
1.2.4 加速度センサによる移動について.....19
1.2.5 ジャイロセンサによる視点移動について.....21
1.2.6 他社(既存の)製品との比較.....23
1.2.7 研究の進行について.....24
第 2 章 機構 （金子担当）.....25
2.1 概要.....25
2.1.1 目的.....25
2.1.2 要求機能・制約条件.....25
2.1.3 全体像・イメージ.....26
2.2 カメラの機構製作.....27
2.2.1 概要.....27
2.2.2 使用部品.....27
2.2.3 タップによる固定方法.....29
2.2.4 NC 旋盤で加工したアルミ板による固定方法.....30
2.2.5 レンズとセンサの固定方法.....31
2.3 遮光壁の機構製作.....34
2.3.1 概要.....34
2.3.2 3D プリンタによる機構製作.....35
2.4 フレームの製作.....36
2.4.1 概要.....36
2.4.2 フレームの製作.....36
2.5 成果.....38
2.5.1 達成具合.....38
2.5.2 映像の映り具合.....39

2.5.3 仕様・重量・寸法 40
2.5.4 考察・評価 40
第 3 章 ステレオ法による距離計測 （橋本担当） 41
3.1 概要 41
3.1.1 目的 41
3.1.2 要求仕様 41
3.2 使用したソフトウェア 42
3.2.1 Visual Studio 2015 42
3.2.2 Open CV 42
3.2.3 Open CV Sharp 42
3.3 レンズ間距離の実験 43
3.3.1 概要 43
3.3.2 原理・理論 43
3.3.3 重要なソースコード 43
3.3.4 実験方法 45
3.3.5 実験結果 46
3.3.6 考察 47
3.4 距離計測法Ⅰ　色抽出法 48
3.4.1 概要 48
3.4.2 原理・理論 48
3.4.3 ソースコード 48
3.4.4 動作確認テスト 50
3.4.5 動作確認結果 51
3.4.6 考察 51
3.5 距離計測法Ⅱ　視差画像法 52
3.5.1 概要 52
3.5.2 原理・理論 52
3.5.3 ソースコード 52
3.5.4 動作確認テスト 53
3.5.5 動作確認結果 53
3.5.6 考察 54
3.6 2 つの方法の比較 54
3.7 距離計測の最終的評価 55
3.7.1 要求仕様との比較 55
3.7.2 考察 55
第 4 章 シミュレーション用ソフトウェアの開発 （片山担当） 56

4.1 概要 56
4.1.1 目的 56
4.1.2 要求仕様 56
4.2 使用したソフトウェア 57
4.2.1 Unity について 57
4.2.2 Visual Studio for Unity 57
4.3 Windows Store App 57
4.4 開発したソフトウェアについて 58
4.4.1 概要 58
4.4.2 動作原理 59
4.4.3 加速度センサ 61
4.4.4 ジャイロセンサ 62
4.5 内蔵カメラの出力実験 63
4.5.1 概要 63
4.5.2 動作原理 63
4.5.3 ソースコード 64
4.5.4 実験方法 64
4.5.5 考察 65
4.6 加速度センサの実験 66
4.6.1 概要 66
4.6.2 ソースコード 66
4.6.3 実験方法 67
4.6.4 実験結果 68
4.6.5 考察 68
4.7 ジャイロセンサの実験 69
4.7.1 概要 69
4.7.2 ソースコード 69
4.7.3 実験方法 70
4.7.4 実験結果 70
4.7.5 考察 71
4.8 OpenCV for Unity について 72
4.8.1 概要 72
4.8.2 3 つのカメラの映像出力 72
4.9 開発したソフトウェアの最終的評価 73
4.9.1 要求仕様との比較 73
4.9.2 考察 73

第 5 章 家具シミュレーションシステムの動作確認 （鈴木担当）..... 74
5.1 概要..... 74
5.2 実験方法..... 75
5.3 実験結果..... 76
5.4 考察..... 78
第 6 章 結論 （須多担当）..... 80
6.1 目標..... 80
6.2 目標への達成度..... 80
6.3 問題点..... 81
6.4 解決策..... 82
6.5 総合評価..... 83
参考文献、URL..... 84
謝辞..... 86

第1章 緒言 （須多担当）

1.1 研究背景

1.1.1 研究テーマ

我々は研究開始当初、K 社から提供された「3D-DEOS ～三次元スキャナを作る～」というテーマに取り組んだ。このテーマは、本のページを読み取るブックスキャナなどに搭載されている DEOS というレンズを 3D に関連する製品に組み込むアイデアを探るというものである。

DEOS は Depth Expansion Optical System の略で、収差制御レンズ(図 1-1)と画像処理のプログラムを組み合わせたものである。収差とは、レンズを通るすべての光が一点に集まらないことによって起きる色ボケや歪みのことである。DEOS は、この収差を制御することで画像内の広い範囲の点にピントを合わせることができる(図 1-2)。

我々は K 社から収差制御レンズが搭載されているカメラの貸与を受け、製品のアイデアをいくつか考えた。しかし、特許の関係から、DEOS の一部である画像処理プログラムの貸与を受けることができなかった。そのため DEOS の要素をテーマから省き、「レンズを使った 3D に関係する次世代製品の開発」というテーマに移行した。

図 1-1 DEOS レンズ

図 1-2 通常のレンズ(左)と DEOS レンズ(右)との焦点距離の比較 [1-1]

1.1.2 3D 技術について

我々はテーマの基となる 3D に関する製品を知るため、以前のテーマの一部にあった 3D スキャナに着目し、調査を行った。3D スキャナとは、ある被写体の形状や大きさを検知し、立体のデータとして取り込むものである。これにより、CAD ソフトウェアで一から 3D モデルを設計しなくても、実際に存在するものから立体のデータを取得することができる。3D スキャナの中には、レーザを照射するもの(図 1-3)や、カメラを複数台使用しているもの(図 1-4)がある。

図 1-3　レーザを照射する 3D スキャナ[1-2]

図 1-4　複数のカメラを使用した 3D スキャナ［1-3］

そして、さらに詳しく調査を行った結果、3D スキャナが使用している原理は多岐に渡り、様々な手法によって被写体を 3D スキャンしていることが分かった(図 1-5)。図中にある「アクティブ」方式は、レーザや超音波を照射するなど、使用者が何らかの動作を被写体に行い、情報を追加することで被写体を 3D スキャンする手法を記している。また、「パッシブ」方式は、カメラで撮影するなど、使用者が情報を追加することなく被写体を 3D スキャンする手法を記している。

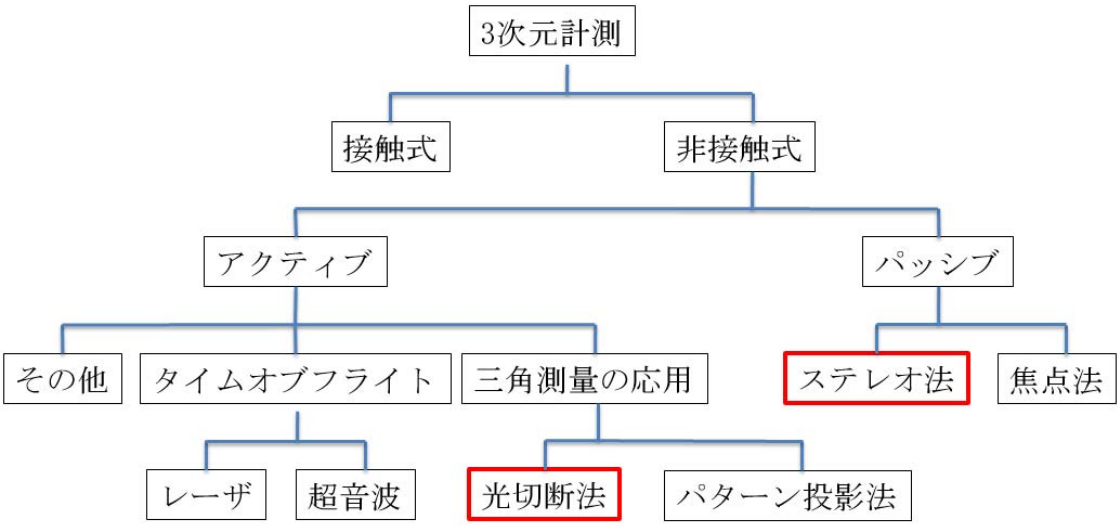

図 1-5　3D スキャンの方法

我々は 3D スキャンについて理解をさらに深めるため、実際にいくつかの実験を行うことにした。3D スキャンを使用できる無償のソフトウェアを調査したところ、赤枠で囲ったレーザを使用することで被写体を 3D スキャンする光切断法と、カメラを 2 台使用することで被写体を 3D スキャンするステレオ法の 2 つの手法のソフトウェアを発見したので、それぞれ試してみることにした。

1.1.3　レーザを使用した 3D スキャン実験

初めに試した実験は、光切断法を用いることで被写体の 3D スキャンを行うものである。光切断法では、まず、三角形の 3 点にカメラとラインレーザ、被写体を配置する(図 1-6)。次に、ラインレーザを被写体に照射し、その形状をカメラで読み取る。こうすることで、ラインレーザを照射した線の歪みの形から被写体の形状を検知することができる。

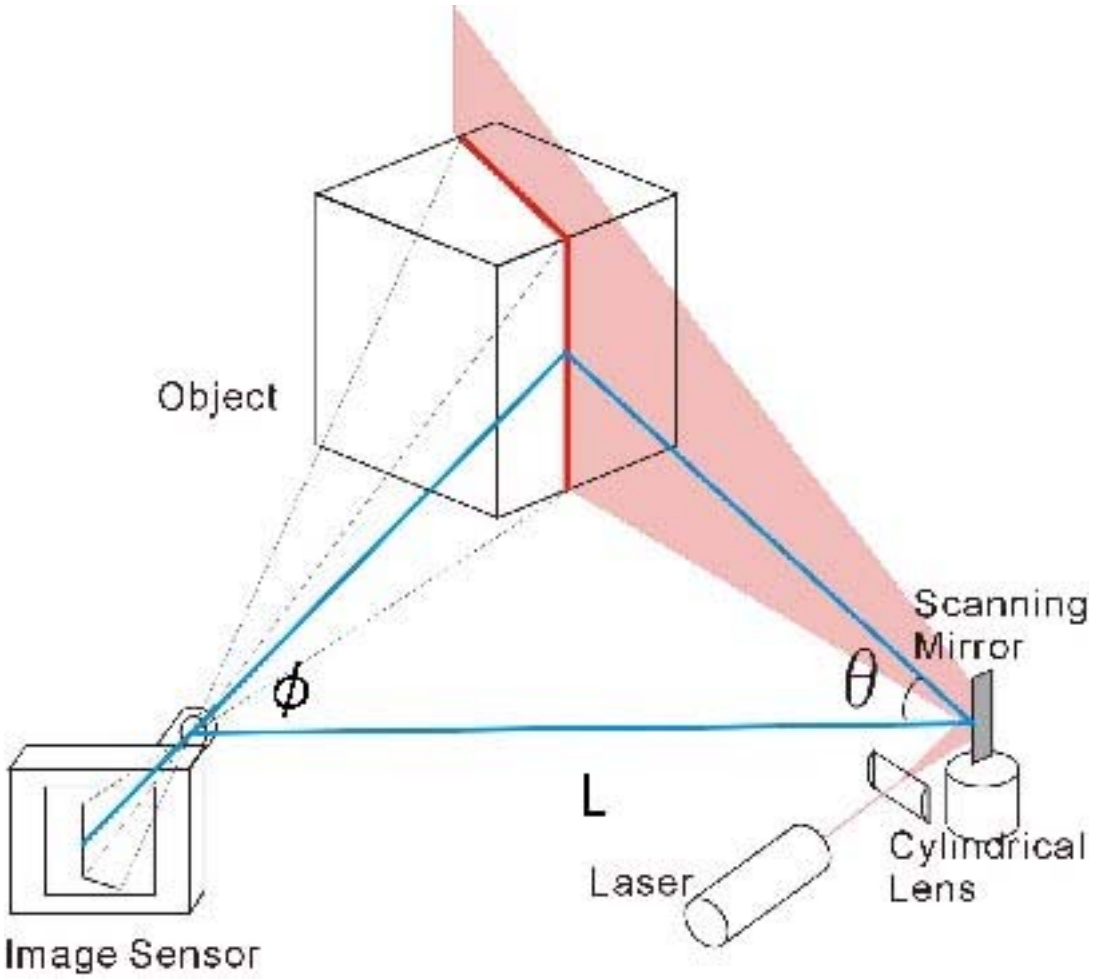

図 1-6　光切断法の原理［1-4］

光切断法の実験を行うにあたり、DAVID という有償ソフトウェアの試用版を使用した(図1-7)。DAVID とは、レーザのゆがみをカメラで読み込み、3D モデルを作成するソフトウェアである。このソフトウェアで作成した 3D モデルは OBJ、STL、PLY の 3 つのフォーマットで保存することができる。

DAVID はラインレーザの色を識別し、それを基に 3D スキャンを行う。そのため、被写体の色がラインレーザと同色の場合や黒など光を吸収してしまう色の場合、さらに被写体が透明でラインレーザが透過してしまう場合でも正常に 3D スキャンができるかどうかを検証する。実験の際はタバコの箱やペットボトル、フィギュアを使用することで被写体の色や形状を変更する。

図 1-7　ソフトウェア DAVID の使用イメージ［1-5］

図 1-8 は DAVID で 3D スキャンを行っている様子を示している。フィギュアの背景には黒点が描かれた紙が見える。これはラインレーザの角度を検知するためのものである。被写体の下部にレーザを照射する際、ラインレーザの角度が下を向きすぎるとレーザ光が背景の用紙からはみ出してしまい、ラインレーザの角度を検知することができなくなってしまう。そのため、被写体を台に乗せてある程度高さを持たせることにした。更に、使用する web カメラについても、被写体と同じ高さを持たせることで映像から被写体がカメラの映像からはみ出してしまうことを防いだ。また、ラインレーザの線が薄くならないように、室内の照明を切った。

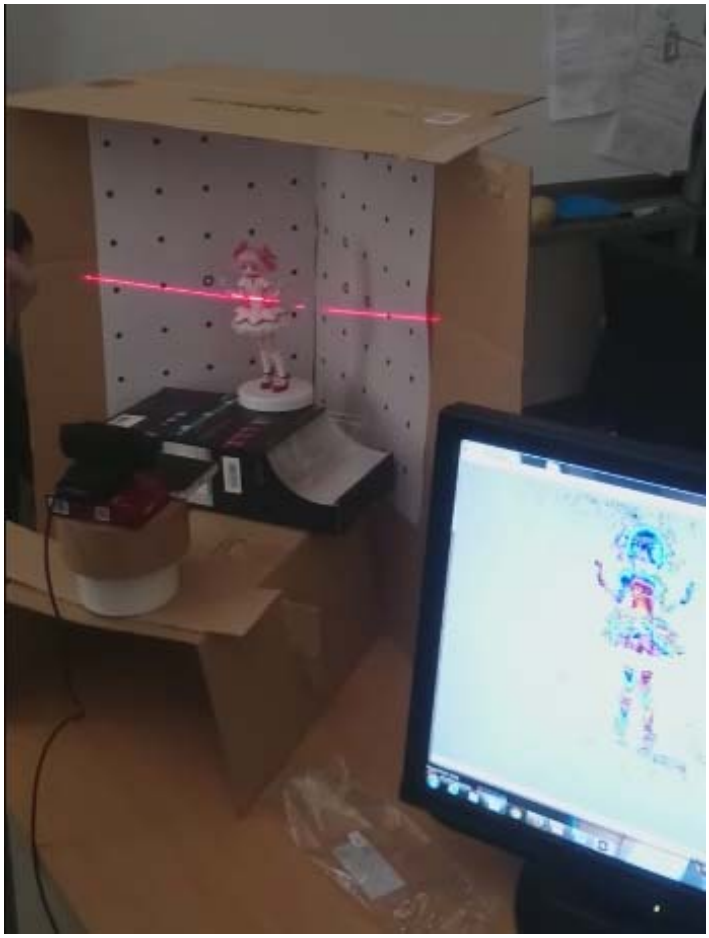

図 1-8 DAVID による 3D スキャン風景

はじめの実験では、フィギュアを 3D スキャンした。図 1-9 はその結果であり、以下の問題点があることがわかった。

1. フュギュア以外の部分も検出している
2. フュギュアの口の部分が検出できていない
3. フュギュアのスカートで隠れてしまった足の付け根の部分が検出できていない

まず 1.について、背景の用紙を段ボールに貼り付けた際、用紙を正確に配置することができていなかった事や、レンズの汚れなどが移りこんでしまったことが原因と考えられる。次に 2.について、DAVID は斜めからラインレーザを照射するため、このフィギュアの顔の構造から鼻の陰に隠れてしまい、口は検出できなかったと考えられる。最後に 3.について、2.と同様に足の付け根部分はフィギュアのスカート部の陰に隠れてしまったことが原因と考えられる。

図 1-9　3D スキャンを行ったフィギュア(左)と DAVID による 3D スキャン結果(右)

また、別の被写体で実験を行ったところ、黒色はラインレーザの光を吸収し線が薄くなってしまうため、また透明な被写体はラインレーザが透過してしまうため 3D スキャンを行うことができなかった。さらに、レーザを使用した 3D スキャンでは、被写体に当たるラインレーザの線の歪みのみを認識しているため、被写体の色を認識することができない、被写体と同じ色のラインレーザは使用することができないことがデメリットとして挙げられる。

1.1.4　カメラを 2 台使用した 3D スキャン実験

本節ではステレオ法という原理を使用して 3D スキャンを行う。このステレオ法とは、三角形の相似の関係を使用して距離を測量する手法のことである(図　1-10)。2 つのレンズと被写体との距離 z は、あらかじめ決められたレンズの焦点距離 f と 2 つのレンズの間の距離 d に加え、被写体を 2 つのカメラで撮影した画像から被写体の像のずれ$|x_1-x_2|$を以下の式に代入することによって算出される。

$$z=\frac{fd}{|x_1-x_2|}$$

z:被写体までの距離
d:レンズ間の距離
f:レンズの焦点距離
x1, x2:画面上の物体の重心

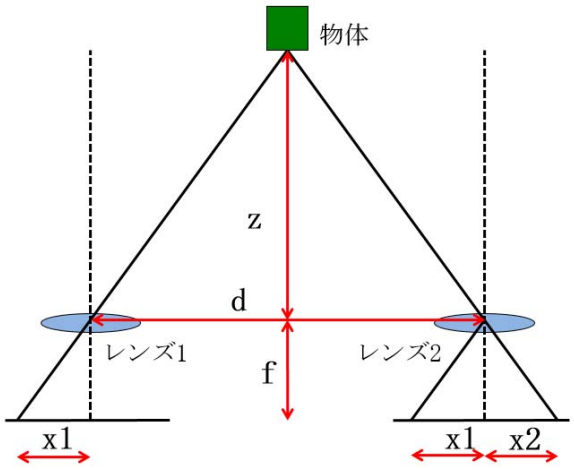

図 1-10　ステレオ法の原理

本実験では 2 つのレンズとして、2 台のスマートフォンのカメラを使用する。さらに、2 枚の写真から 3D モデルを構成するために、Agisoft StereoScan(以下、StereoScan)というソフトウェアを使用した(図 1-11)。StereoScan で作成した 3D モデルは PDF、OBJ、SLY など 8 つのフォーマットに保存することができる。

なお、本実験では、2 枚の画像の解像度などの条件を揃えるために、使用するスマートフォンを同機種の 2 台とした(図 1-12)。階段を被写体とし、メジャーで階段とスマートフォンとの距離を測定した後、スマートフォンを 2 台平行に並べ被写体を撮影する。更に、3D モデルを作成することができる条件を知るために、スマートフォンの間の距離や被写体までの距離を変えて実験する。

図 1-11　ソフトウェア Agisoft StereoScan の使用イメージ [1-6]

図 1-12　実験で使用したスマートフォン(SC-04D)2 台

実験の結果、図 1-13 の 2 枚の画像を StereoScan で合成すると、階段の 3D モデル(図 1-14)を作成することができた。これは 3D モデルを閲覧できる特殊なフォーマットの PDF で出力されており、PDF 内で自由に被写体の向きを変更し閲覧することができる。しかし、大きく向きを変更すると、3D モデルの形が歪んでしまう。

図 1-13　3D モデルの作成成功時の 2 枚の画像

図 1-14　StereoScan による合成結果

また、カメラ間の距離を d、被写体までの距離を z とし、二つの距離をそれぞれ変更した際の結果である(表 1-1)。〇が 3D モデルの合成に成功、×が失敗を表している。

表 1-1　二つの距離を変えたときの 3D モデルの合成結果

z[mm]＼d[mm]	70	100	130
2000	×	×	×
2500	〇	〇	×
3000	〇	〇	×

以上の結果から、レンズ間距離が短くなければ 3D モデルの作成ができないことが分かる。理由として、レンズ間距離が長すぎると 1 つの被写体がそれぞれの画像で別の被写体と認識され、3D モデルを作成するべき被写体が存在していないとソフトウェア内で認識されることが挙げられる。また、レンズ間距離が短かったとしても、被写体までの距離が近い場合、2 枚の画像に写る被写体の位置の差が大きくなるため、これも同様の理由で 3D モデルを作成することができない。これらのことから、2 つのカメラを使用して 3D スキャンする場合、レンズ間距離と被写体までの距離の関係が重要であることが分かった。

1.2 研究概要

1.2.1 製作物の決定

我々は前節で述べた二つの実験を踏まえ、カメラを 2 台使用した 3D スキャンについて研究を進めていくことにした。その理由として、レーザを使用した 3D スキャンでは、レーザが人の目に入った場合に危険であることや被写体の色を認識することができない点が挙げられる。これに対してカメラを 2 台使用した 3D スキャンでは、2 つのカメラの正確な位置決めをする必要はあるものの、使用する上で危険がなく、被写体の色を認識することもできるためである。

そして、ステレオ法で距離を算出することができる点に着目し、カメラ 2 台(以下、ステレオカメラ)を使って間取り測量のアイデアを考えた(図 1-15)。考案した間取り測量とは、ステレオカメラで壁や天井、あるいは家具と家具の間の距離などを測量するものである。これのメリットは、メジャーなどを使用せずに、手軽に距離を測量できる点である。

図 1-15　間取り測量のアイデア［1-7］

しかし、間取り測量について調査したところ、既存の製品にはレーザを使用し、容易に間取りを測量することが可能なものがあることがわかった。そこで、我々はこの間取り測量というアイデアから、室内で距離を計測する点のみに着目した。そして、この計測した距離を用いて、タブレット PC が室内のどこにあるのか推定できると考えた。これに AR 技術を組み合わせることで、室内に家具の CG を配置し、家具の配置についてのシミュレーションを行うというアイデアに至った（図 1-16）。

図 1-16　家具のシミュレーションを行っている風景[1-8]

AR 技術とは、カメラから取得した映像に物の画像や CG を合成するものである。また、AR 技術の中には、マーカ AR とマーカレス AR がある。

マーカ AR とは、マーカという基点を実際に用意し、カメラでマーカを写すことにより、合成する CG の位置や向きを取得するというものである(図 1-17)。

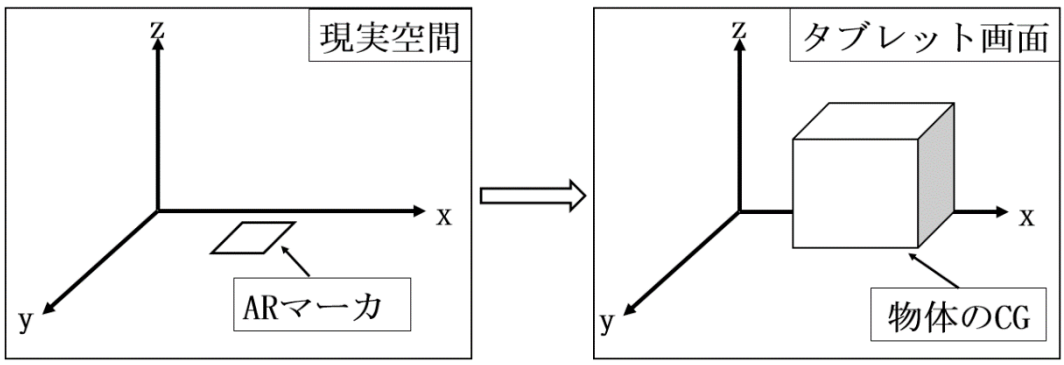

図 1-17　マーカ AR の概要図

一方、マーカレス AR とは、マーカを用意せずに合成する CG の位置を取得するものである。マーカレス AR の例として、GPS を使用し使用者の現在位置を取得することで、その周辺に対して物の画像や CG などを合成するものなどがある(図 1-18)。

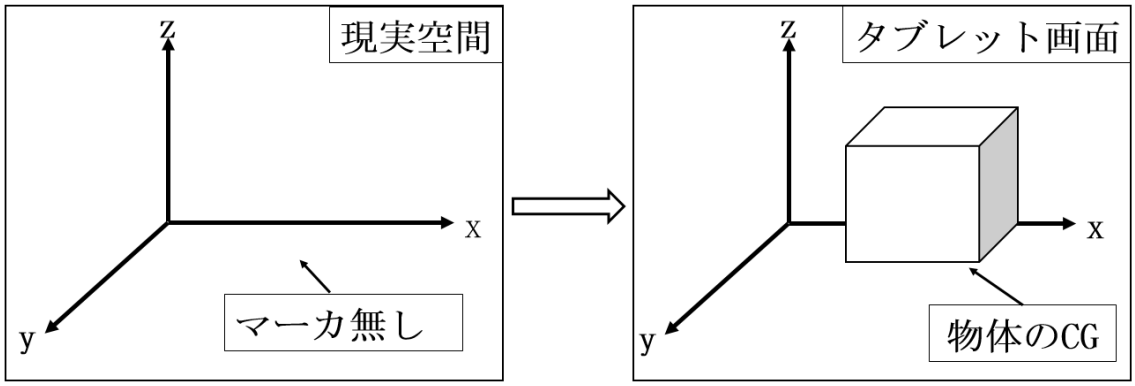

図 1-18　マーカレス AR の概要図

我々が開発する AR 技術を使用した家具のシミュレーションシステムは、マーカを使用せず、ステレオカメラなどで使用者の位置と向きの推定を行うので、このマーカレス AR に該当する。

1.2.2 家具シミュレーションについて

本研究での目標とする家具シミュレーションシステムは、室内で実際の家具を動かすことなく、AR 技術を使用して家具の CG を自由に動かすことにより、室内の景観の変化を模擬できるシステムである。タブレット PC 用のソフトウェアとして開発し、タブレット PC のカメラを使用して取得した室内の映像に家具の CG を合成することで実現する。これの用途として、モデルルーム内で新居の見学に来た人が使用するという場面を想定している。

また本研究における家具シミュレーションシステムは 2 つの要素から成り立っている。1 つ目は、室内のどこに家具を配置したいかを設定することである（図 1-19）。

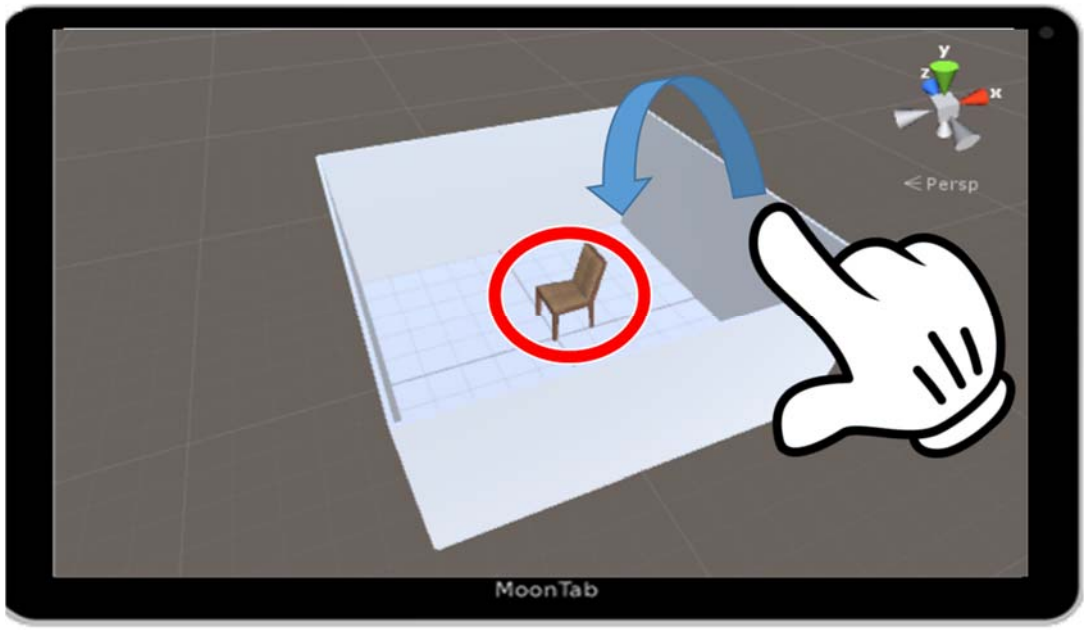

図 1-19　家具の CG の位置を決める際のイメージ

2 つ目は、室内で実際にソフトウェアを起動し、タブレット PC を通して配置した家具の CG を見ることである(図 1-20)。ソフトウェアを起動すると、タブレット PC に内蔵されているカメラ(以下、内蔵カメラと称する)からの映像が画面上に出力される。そして、図 1-19 で設定した場所に内蔵カメラを向けると、室内の映像と家具の CG が合成され、実際に家具が存在するかのような光景を体験できる。また、タブレット PC を持ったまま移動することで、家具の CG に近づくことや角度を変えて見ることができる。

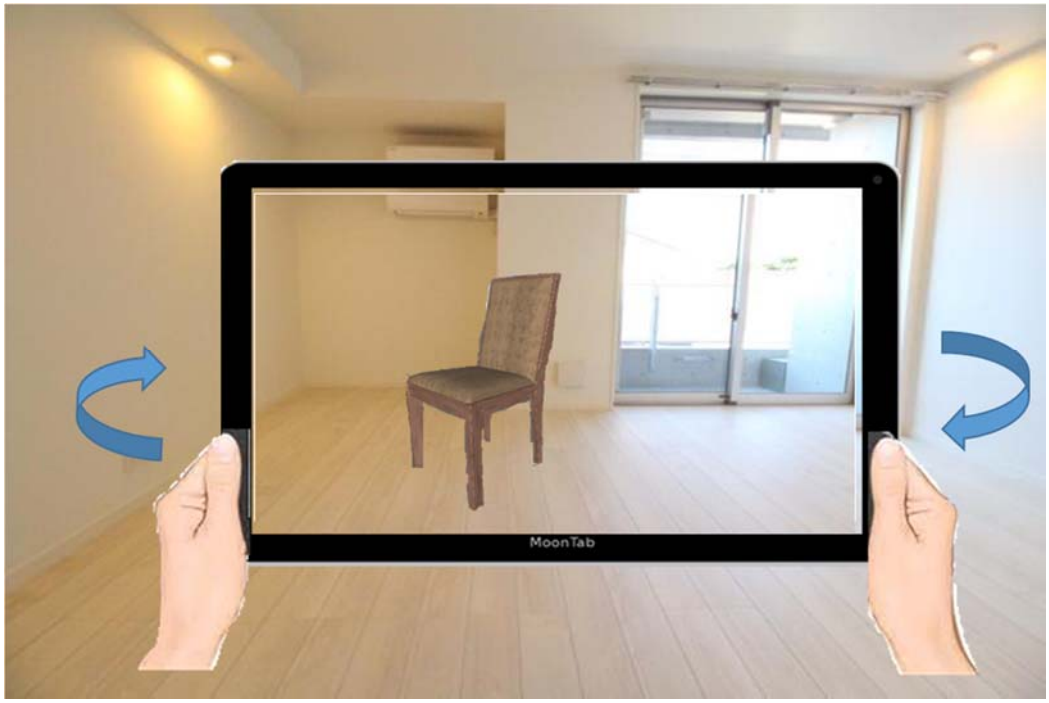

図 1-20　家具シミュレーションシステムを動作させた際のイメージ

本研究における家具シミュレーションシステムを実現するには、使用者の自己位置と向きの推定が必要となる(図 1-21)。ここで述べる自己位置と向きとは、室内のタブレット PC の位置と向きのことを指す。この自己位置や向きを推定することで、タブレット PC を自由に持ち歩き、家具の CG を様々な位置や方向から見ることができるようになる。

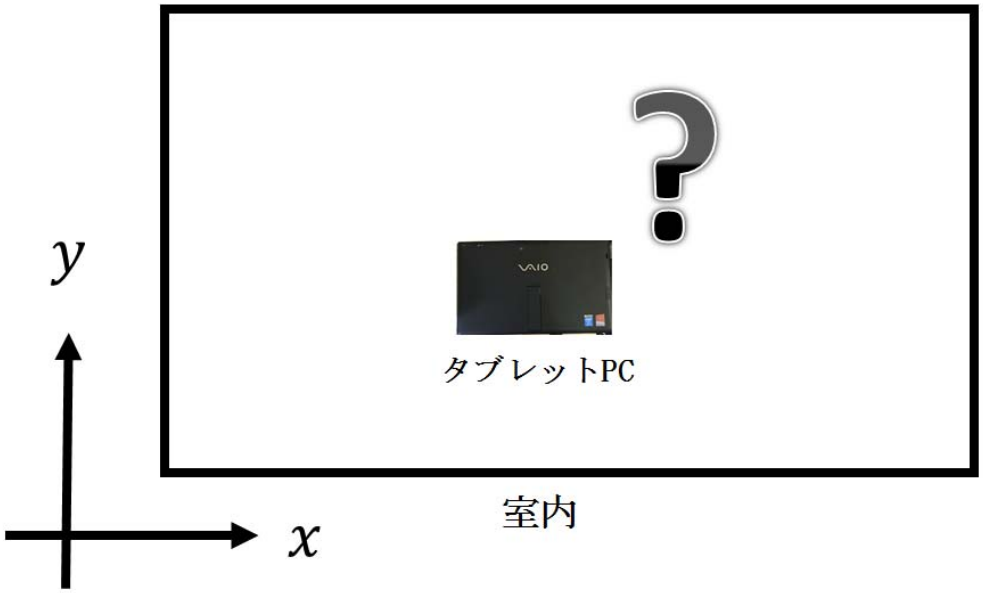

図 1-21　室内における自己位置の把握

本研究では、次の 3 つの技術を用いて自己位置と向きの推定を行う。

1. ステレオ法による距離計測
2. 加速度センサによる移動距離の算出
3. ジャイロセンサによる向きの取得

1.2.3 ステレオ法について

本節では 1 つ目の技術であるステレオ法による距離計測について解説する。ステレオ法とは、1.1.4 で解説したようにステレオカメラを使用して距離を計測する手法のことである。また、本研究において定義するステレオカメラとは、カメラを 2 台平行に並べたものを指す。このステレオカメラをタブレット PC に取り付けて、室内の壁までの距離を測量することで自己位置の推定を行う(図 1-22)。

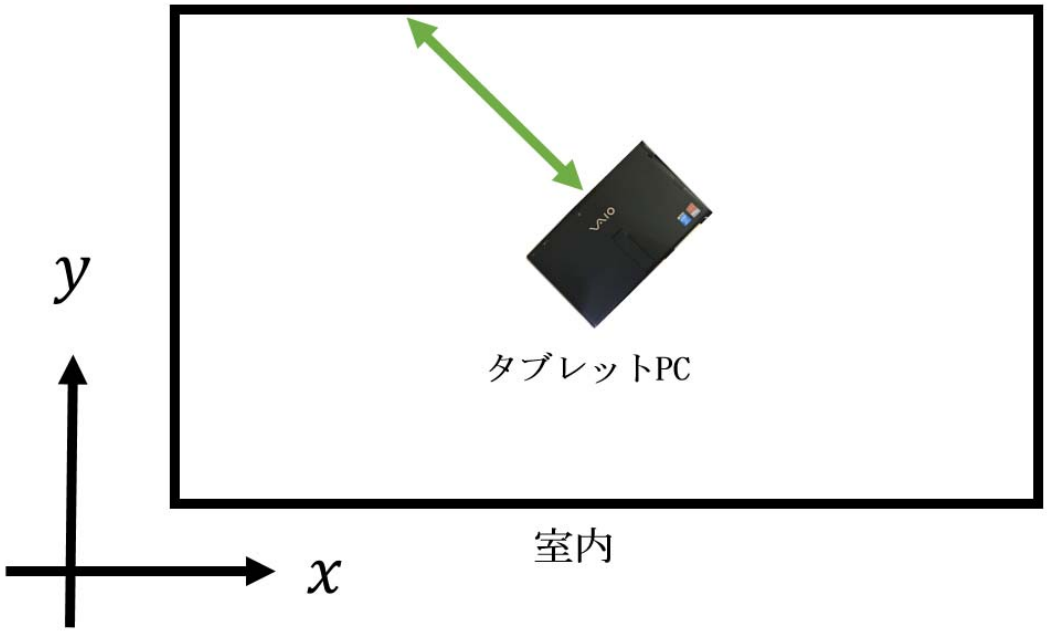

図 1-22　ステレオカメラによる距離計測

1.2.4 加速度センサによる移動について

本項では、加速度センサによる移動距離の算出について解説する。加速度センサとは、被写体が移動した場合、その移動の加速度を検出し数値として出力するセンサのことである。PC の HDD ドライブを保護するための衝撃検出や、重力加速度を検出し携帯電話の画面表示の向きを変える縦横検出などに用いられている。

この加速度センサの加速度を検出する原理は次のようになっている。まず、加速度センサの内部はバネに重りがつるされている構造となっている。この構造により、被写体が移動した時の重りの移動量とバネ係数、重りの重さをニュートンの法側に代入することで加速度を算出することができる(図 1-23)。

加速度センサが検出した値を 2 階積分することにより、距離を算出することができる。

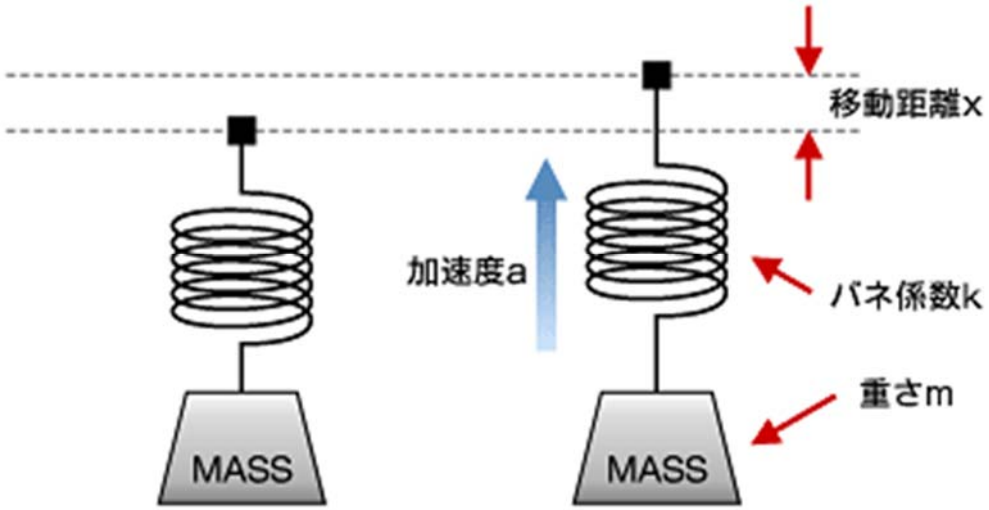

図 1-23　加速度センサの検出原理［1-9］

この加速度センサを用いることで、タブレット PC の室内での移動距離を算出し、自己位置の推定の補助を行う(図 1-24)。前節で述べたステレオカメラでは、カメラの向いた方向のみ距離を算出することはできるが、その他の軸方向については距離を算出することができない。これに対して、加速度センサは 3 つの軸方向の距離を算出することができる。しかし、加速度センサはその構造上、常に加速度の値が振動し続けてしまう上、積分すると積分による誤差が生じる。また、位置を逐一更新できるステレオカメラと違い、加速度センサでは加速度を積分することで算出した誤差を含む移動距離を累積させていくことで自己位置の推定を行う。そのため、誤差も累積してしまい、使用時間に応じて自己位置の推定の精度が落ちてしまう可能性がある。以上のことから、加速度センサはあくまでステレオカメラを取り付けていない軸方向への移動を補完するために使用する。

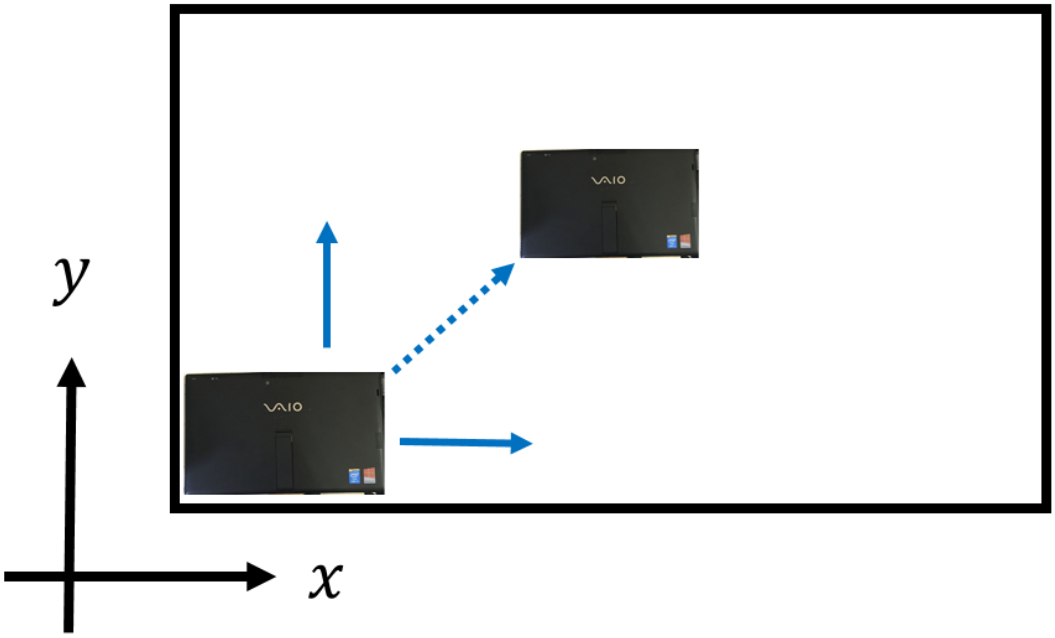

図 1-24　加速度センサによる移動距離の算出

1.2.5 ジャイロセンサによる視点移動について

本節ではジャイロセンサによる向きの推定について解説する。ジャイロセンサとは、被写体が回転移動した場合の角速度を検出し、数値として出力するものである。これは主にロボットの姿勢制御のためやデジタルカメラの手振れ補正で用いられる。また、最近ではスマートフォンやタブレット PC に内蔵されており、ゲームやソフトウェアの一部にも使用されている。

ジャイロセンサが角速度を検出する原理は、被写体が回転した時にセンサの素子を振動させることで発生したコリオリの力を解析し角速度を算出するものが主流である(図 1-25)。コリオリの力とは、移動している被写体が回転した時に生じる、移動方向と回転軸に直交する力である。

ジャイロセンサが出力した角速度の値を積分することによって角度を算出することができる。

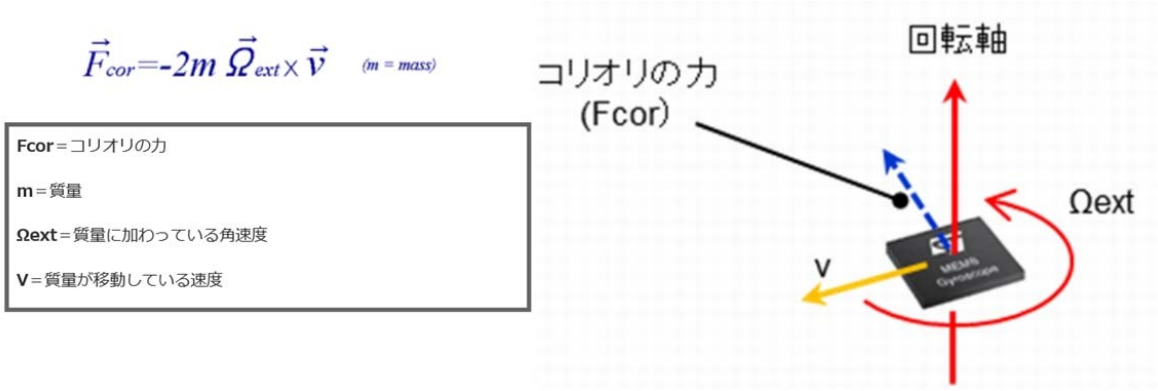

図 1-25　ジャイロセンサの検出原理［1-10］

以上のことから、ジャイロセンサを用いることで、室内であってもタブレットPCの向きを推定することができると考えられる(図 1-26)。

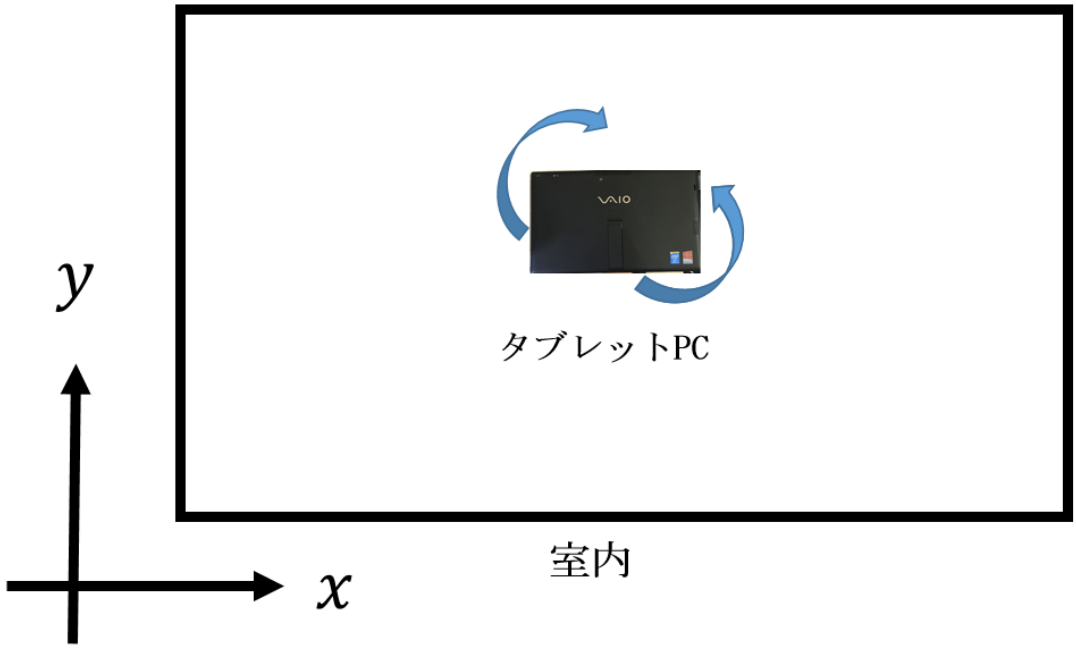

図 1-26　ジャイロセンサによる向きの推定

1.2.6 他社(既存の)製品との比較

本節では、我々が開発する家具シミュレーションシステムを既存製品と比較する。図 1-27 は家具やインテリアの商品を販売している IKEA が公開しているスマートフォン用ソフトウェアである。まず、このソフトウェアで IKEA のカタログ内のページに載っている商品の情報を読み取る。その後カタログを閉じると、カタログの上に家具の CG が出現し、我々と同様に家具の配置を模擬することができる。図 1-27 はカタログの上に CG の机が表示されている様子である、ただし、カタログ自体は「Show products」という文字に隠れてしまっている。これはカタログをマーカとして使用しているという意味でマーカ AR の技術に分類される。

図 1-27　IKEA のソフトウェア［1-11］

IKEA のソフトウェアの欠点は、スマートフォンのカメラに基点となるカタログが写っている必要がある点である。そのため、天井や壁の高所のような基点の配置が困難な位置への家具の配置ができない。また、大きな家具の CG を見る際、その上部までカメラの角度を上げると基点がカメラの外に出てしまい、家具の CG が消えてしまうようこともある。

これに対して、我々が考えている家具シミュレーションシステムは、基点となるマーカが存在しないマーカレス AR である。マーカの代わりに、ステレオカメラなどで使用者の位置と向きを推定し、事前に決めた家具の CG の位置と照らし合わせてシミュレーションを行うため、先に述べた基点の配置が困難な位置や、見上げた際に基点がカメラから出てしまう程大きな家具の CG であってもシミュレーションを行うことができるというメリットがある。

1.2.7 研究の進行について

本研究を実現するにあたって必要な要素を検討し、その結果 3 つの班に分かれ作業を分担した。

1 つ目の班を機構班とする。機構班の役割は以下の 4 つである

1. 前述したステレオ法を実現するためのステレオカメラの製作
2. ステレオカメラ製作のための、レンズとカメラの基板の距離が調節できる機構の実現
3. レンズとカメラ基板の間に光が入らないようにすること
4. 製作したステレオカメラのタブレット PC への取り付け

2 つ目の班を画像処理班とする。画像処理班の役割は以下の 2 つである。

1. ステレオ法を使用するのに最適な二つのカメラのレンズ間距離の算出
2. ステレオ法を用いたレンズと被写体までの距離の測量

3 つ目の班をソフトウェア開発班とする。ソフトウェア開発班の役割は以下の 3 つである。

1. タブレット PC 内蔵カメラからの映像の取得と画面上への出力
2. 取得した映像と家具の CG の合成
3. タブレット PC を持ち歩いた際の現在位置のリアルタイムな取得

上記の 3 班が分担した役割について研究を進め、家具シミュレーションシステムの完成を目指す。

第2章 機構 （金子担当）

2.1 概要

2.1.1 目的

本章では、本研究で製作する家具シミュレーションシステムに必要なカメラフレームについて解説する。カメラフレームの製作では、家具シミュレーションシステムの実現のために必要な、カメラと、製作したカメラをタブレットに取り付けるためのフレームの製作を目的とする。また、カメラは、レンズとカメラ基板、遮光壁などを組み合わせて製作するが、それをタブレットに固定する仕組みについても解説する。

2.1.2 要求機能・制約条件

家具シミュレーションシステムの製作にあたって必要な要求性能・制約条件を以下に示す。

1. 前述したステレオ法を用いるために、2 つのカメラを固定して、ステレオカメラを製作すること
2. カメラ製作のための、レンズとカメラ基板の距離を調節する仕組みを実現すること
3. レンズとカメラ基板の間に光が入らないようにすること
4. 製作したカメラをタブレットに取り付けること

これらを考慮し製作に取り組んで行く。

2.1.3 全体像・イメージ

下図　2-1 は、要求性能・制約条件を元に 3 次元 CAD で設計したカメラフレームの全体図である。タブレット PC が 2 枚のアルミ板で挟まれた構造をしており、背面に 2 つのカメラ(ステレオカメラ)が取り付けられている。それぞれのカメラはレンズ、カメラ基板、遮光壁がアルミパーツを介して固定されて作られている。さらに、タブレットを挟むアルミ板は、前面はディスプレイ閲覧用、背面は軽量化用と各種センサ用の穴が開けられている。安価である事と、加工し易さを考慮し、部品は全てアルミを使用する。また、設計図を製作する際には、3 次元ソフトウェアである SolidWorks を使用する。次節より、部品ごとの詳しい説明をしていく。

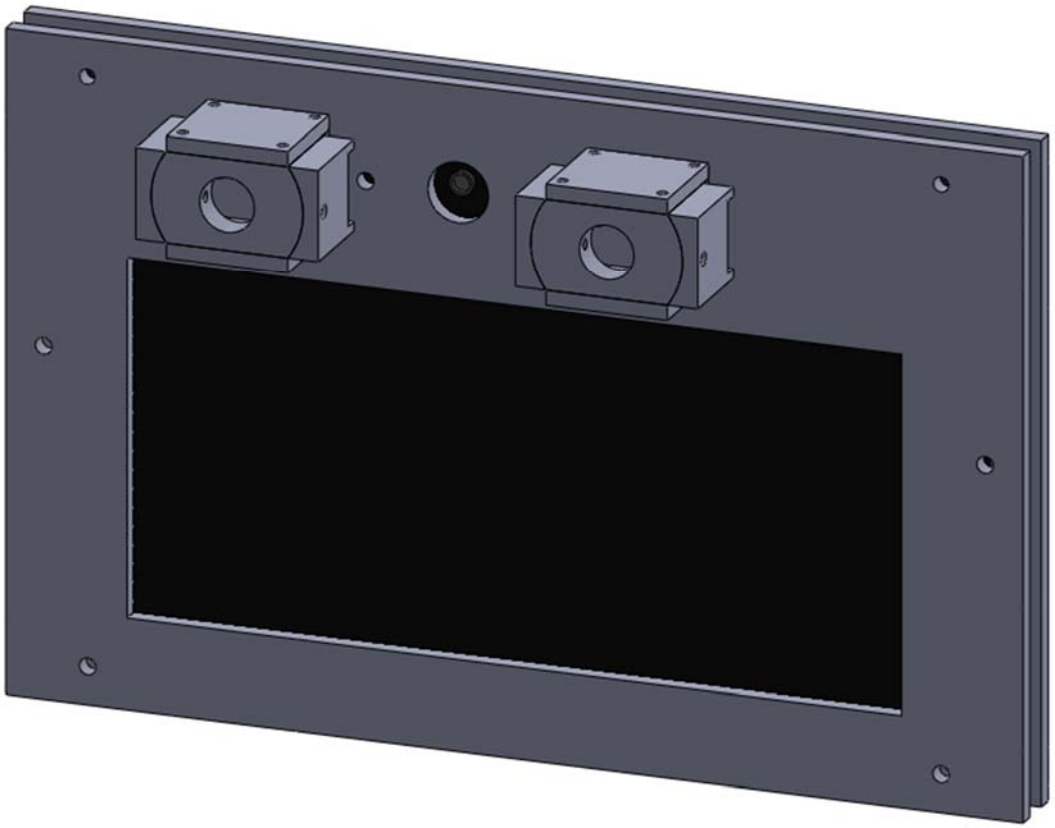

図 2-1　設計した家具シミュレーションのカメラフレームの CAD 図

2.2 カメラの機構製作

2.2.1 概要

家具シミュレーションシステムを実現させるには、ステレオ法を用いた距離測量を行うため、2 台のカメラが必要となる。本節では、その 2 台のカメラの製作法について説明する。

2.2.2 使用部品

本研究では、K 社から提供して頂いたレンズと、Logicool 製のウェブカメラ(HD Pro web camera C920)を分解し、カメラ基板部分のみを使用する。K 社から提供して頂いたレンズには、M17.5×P0.5 という JIS 規格に無い特殊なサイズのネジ山が切られている(図 2-2 赤丸部分)。カメラ映像のピントを合わせるためにレンズとカメラ基板との間の距離が調節可能である必要があるが、そのためにこのネジ山を利用する。

図 2-3 は今回使用するウェブカメラを解体して取り出したカメラ基板である。赤丸で示してあるのがセンサ部分となる。また、センサ部分の左右にφ3.5 の穴があり、それを活用して後に述べるアルミパーツ上にカメラ基板を固定することになる。

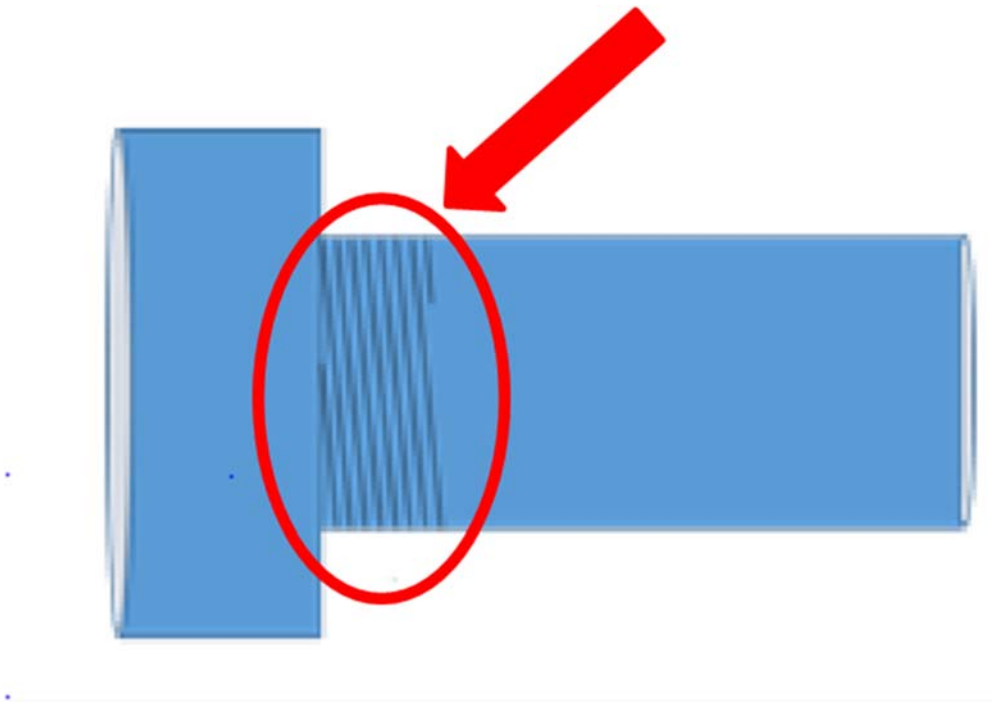

図 2-2　使用するレンズのイメージ図

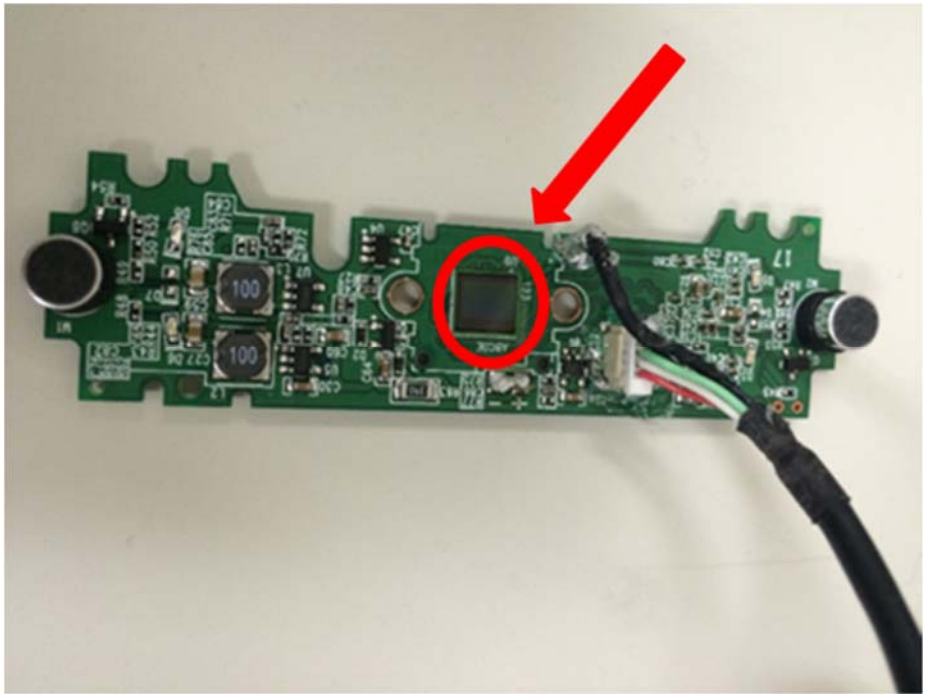

図 2-3　解体したウェブカメラの基板部分

2.2.3 タップによる固定方法

既に述べたようにレンズの固定及びカメラ基板との距離調節には、レンズに切られたネジ山を利用する。そのためには、それに対する雌ネジが必要である。しかし、JIS 規格には、M17.5 というサイズの規格は存在しない。そこで、市販の M18×P0.5 のハンドタップを購入し、φ17.3 やφ17.5、φ17.7 のアルミ板の下穴にタップ加工してレンズのネジ山が使用可能か試してみた。図 2-4 は実際に加工したネジ穴であるが、全くレンズのネジ山と噛み合わなかった。そのため、既存のハンドタップで M17.5×P0.5 と噛み合うネジを切るのは難しいと考えた。

図 2-4　M18×P0.5 のハンドタップによるネジ穴

2.2.4 NC 旋盤で加工したアルミ板による固定方法

ハンドタップを使用して M17.5×P0.5 ネジ穴を製作できなかったため、次に工学院大学八王子キャンパスの 19 号館にある NC 旋盤を利用することにした。円状のアルミ板に旋盤を利用して M17 の下穴を開け、その後、NC 旋盤で M17.5×P0.5 のネジを切った。図 2-5 のように NC 旋盤で切ったネジ穴に対してレンズをねじ込むと、ネジが噛み合うことがわかった。これにより、レンズのネジ山を活用したカメラ基板との距離調節が可能となった。

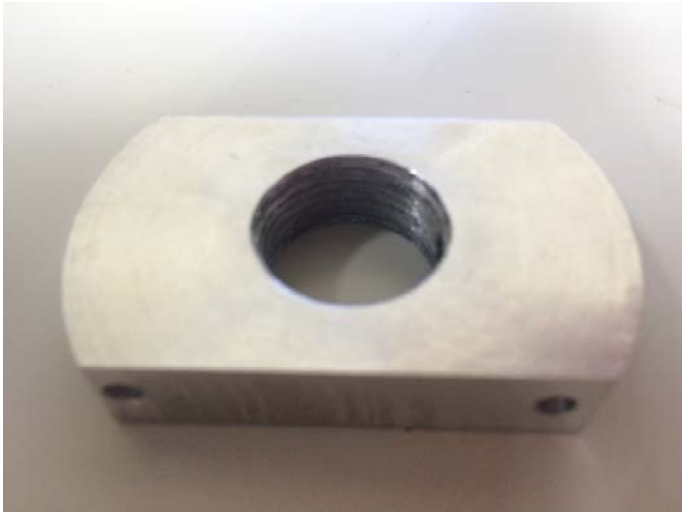

図 2-5　NC 旋盤で加工した M17.5×P0.5 のネジ穴

2.2.5 レンズとセンサの固定方法

M17.5×P0.5 のネジ穴の製作が成功したため、次にレンズとカメラ基板を固定するためのアルミパーツの製作を行った。まず、SolidWorks で設計を行った。図 2-6、図 2-7、図 2-8 が、部品ごとの設計図であり、それぞれ、前面、背面、上下面のパーツを表している。

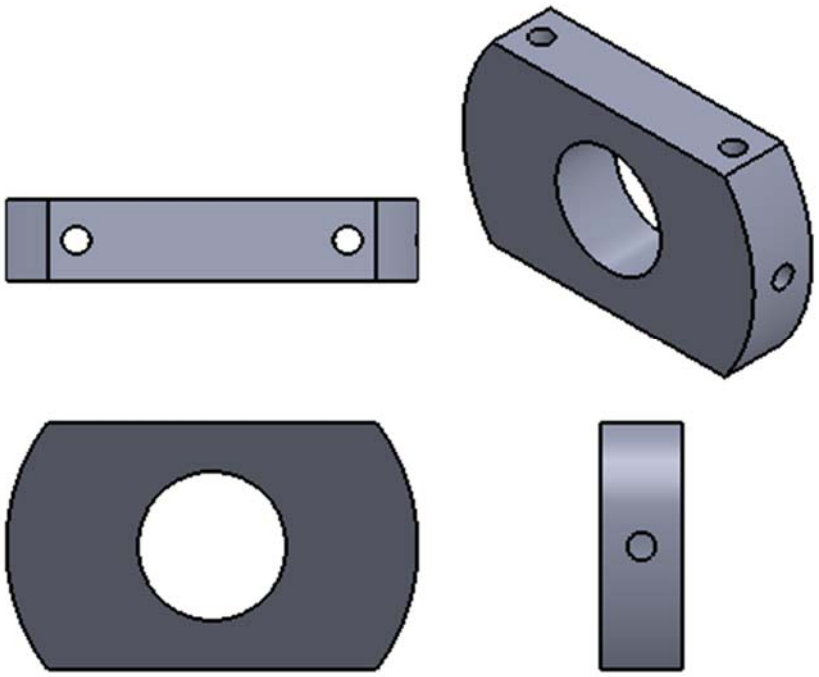

図 2-6　レンズ固定用の部品(前面)

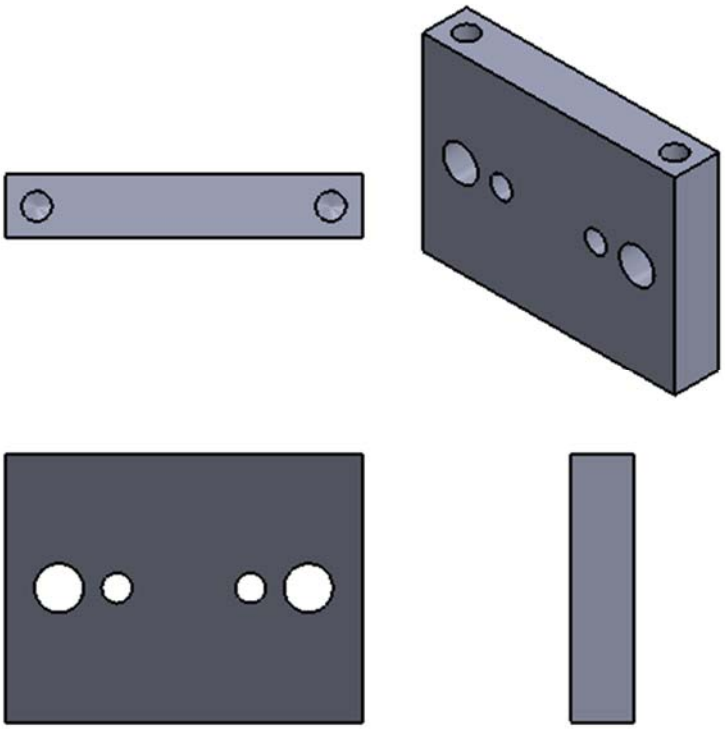

図 2-7　基板を固定する背面用の部品

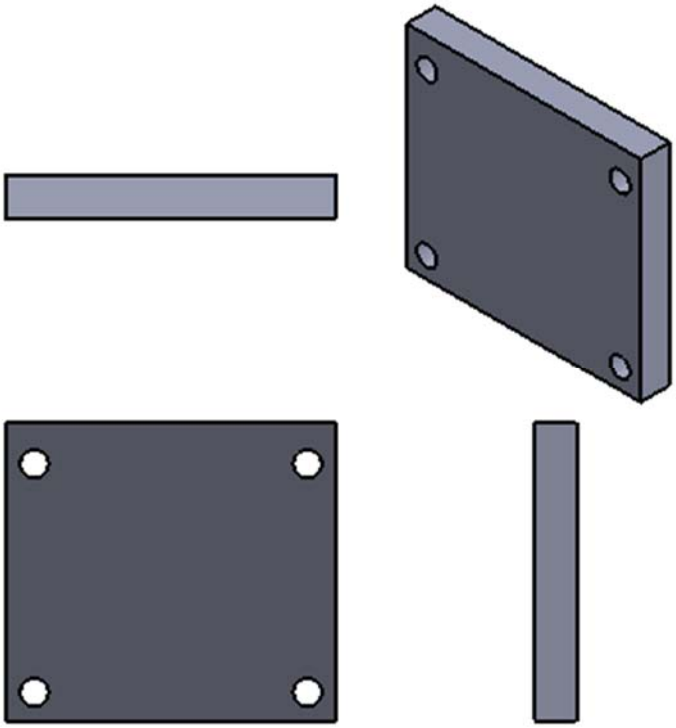

図 2-8　上下用面の部品

本節より、「レンズをねじ込むM17.5×P0.5のネジ穴のある部品」を「レンズ固定用の部品」、「レンズ固定用の部品と基板を固定する背面用の部品を繋げる部品」を「上下面用の部品」と称する。組み立て図が図 2-9 である。

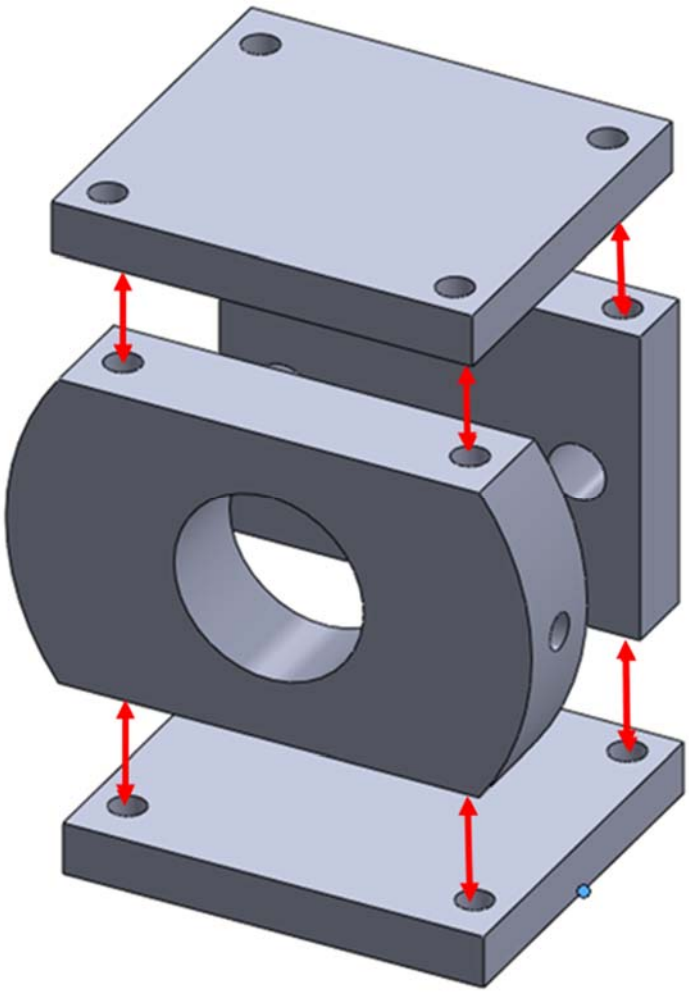

図 2-9　カメラの組み立て図

2.3 遮光壁の機構製作

2.3.1 概要

前節により、レンズとカメラ基板を固定したカメラが完成した(図 2-10)。しかし、USB ケーブルで PC に接続して映像を映したところ、レンズとカメラ基板の間に光が入ってしまい(図 2-10 赤矢印)、映像が淡くなってしまうことが判明した(図 2-11)。そのため、レンズとカメラ基板の間に光が入らないようにする必要が出た。そこで、カメラ側面から入る光を遮断する遮光壁を製作することにした。

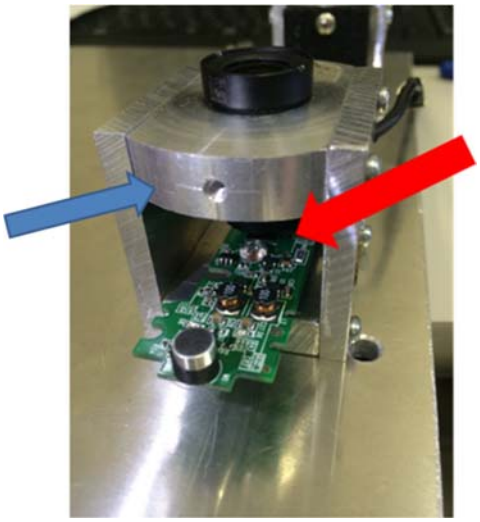

図 2-10　遮光壁取り付け前の図

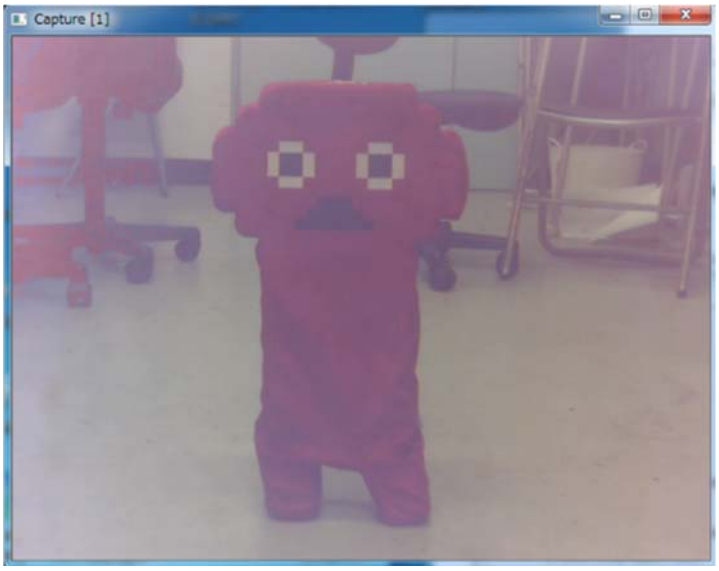

図 2-11　映像のキャプチャ

2.3.2 3D プリンタによる機構製作

前ページの図 2-10 の青矢印で示した部分のように、遮光壁は曲面を持った構造でなければならない。そのため、3D プリンタを用いて製作することにした。図 2-12 は、SolidWorks を用いた部品の設計図である。赤丸で示した部分は、カメラ基板に繋がるコードを通すための穴、青丸で示した部分は、レンズ固定用の部品にネジで固定するための穴である。図 2-13 は製作した遮光壁をカメラへ取り付ける組み立て図である。

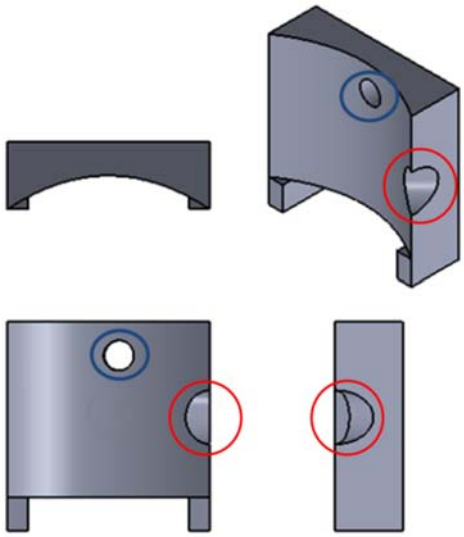

図 2-12　遮光壁

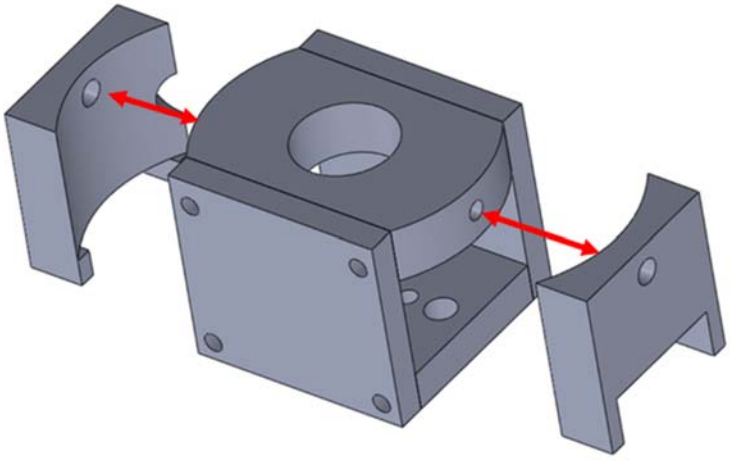

図 2-13　カメラへの遮光壁の取り付け

2.4 フレームの製作

2.4.1 概要

家具シミュレーションシステムは、タブレットを持ち運んで使用することにより実現されるため、製作したカメラをタブレットに取り付ける必要がある。本節より、製作したカメラをタブレットに取り付ける土台のことを、フレームと称し、その詳細を解説する。

2.4.2 フレームの製作

フレームの製作には、カメラと同様にアルミ板を使用した。図 2-14 と図 2-15 は、SolidWorks を用いた部品ごとの設計図である。図 2-14 はタブレットの液晶側の設計図である。赤丸で示した凹みの部分はタブレットのセンサを塞ぐことを避けるためのものである。図 2-15 はカメラを固定する側の設計図である。フレームに切ったネジ穴を使用し、前述のカメラ基板を固定する部品の穴からネジを通してカメラを固定する。タブレットの液晶側フレームには、内側に厚さ 1mmのゴム板を挟みタブレットが滑らないようにした。また、図 2-16 はカメラとフレームの組み立て図である。

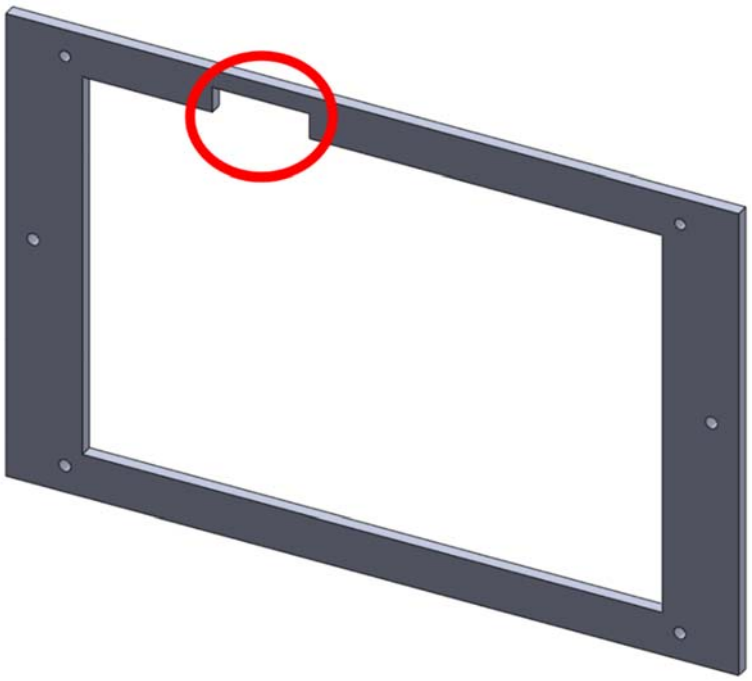

図 2-14　タブレットの液晶側フレーム

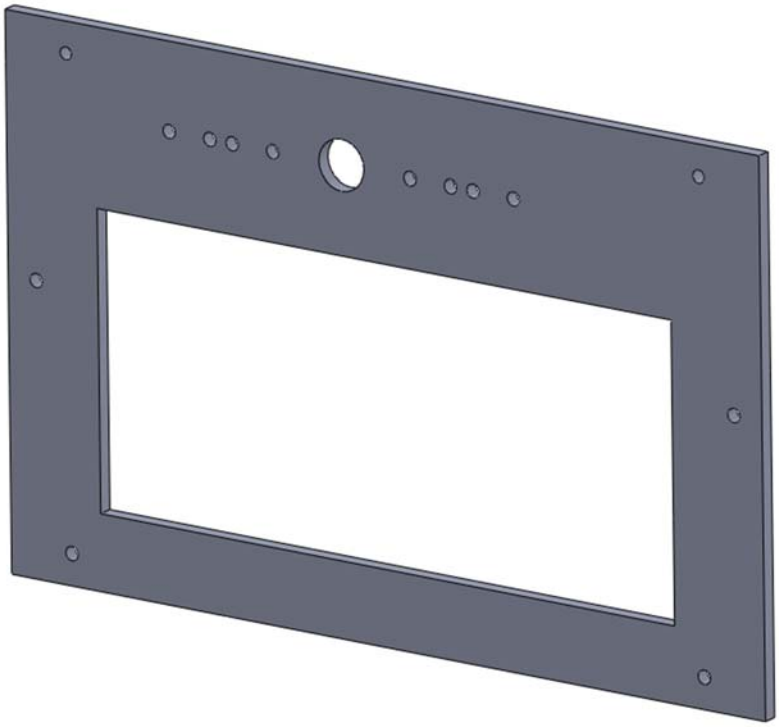

図 2-15　カメラ固定側のフレーム

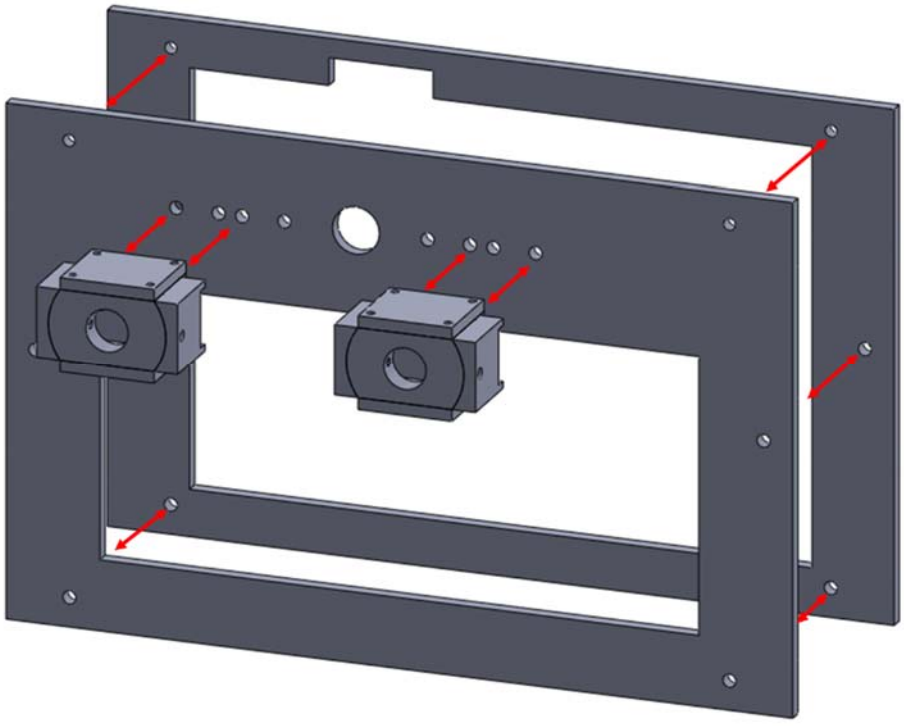

図 2-16　カメラとフレームの組み立て図

2.5 成果

2.5.1 達成具合

図 2-17、図 2-18 は完成させたカメラフレーム全体の写真である。

図 2-17　完成したカメラフレームの写真(背面のステレオカメラ側)

図 2-18　完成したカメラフレームの写真(液晶側)

2.5.2 映像の映り具合

本章で製作したカメラによる映像の例を紹介する。カメラを PC に接続し、PC 上で映像をキャプチャした。図 2-19(左)のように遮光壁を付けなかったときは映像が紫がかってぼやけているが、遮光壁を取り付けることで図 2-19(右)のように鮮明な映像が得られた。図 2-20(左)が焦点調節前の映像である。それに対し、焦点距離調節後の図 2-20(右)は鮮明な映像が得られていることがわかる。これにより、2.2.4 で製作したレンズ固定用の部品のネジ穴が機能し、レンズとカメラ基板との距離が調節できていることがわかる。

図 2-19　遮光壁がない場合の映像(左)とある場合の映像(右)

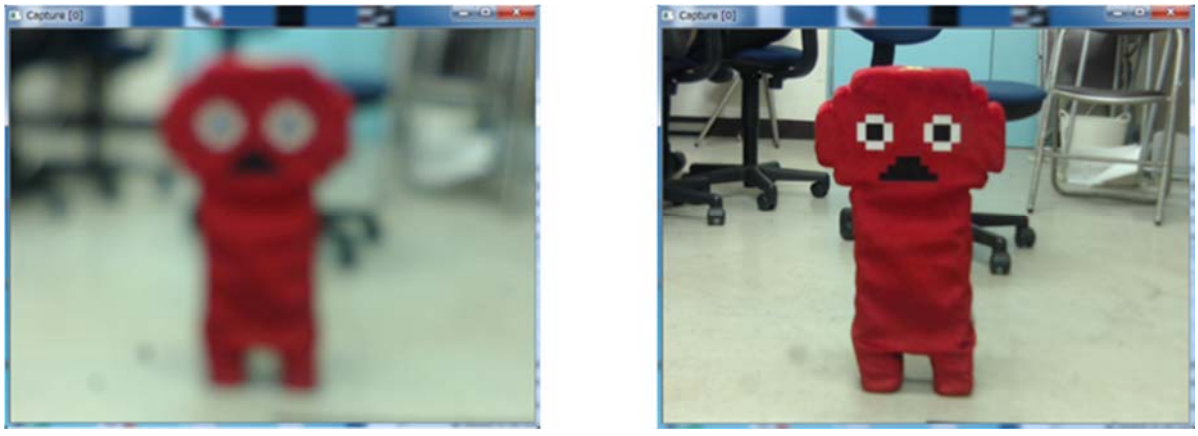

図 2-20　焦点調節前の映像(左)と焦点調節後の映像(右)

2.5.3 仕様・重量・寸法

カメラ一つの重量が 168g、タブレットの重量が 758g、フレームの重量が 914g、総重量が 2008g となった。寸法は図 2-21 にも示した通り、縦寸 218mm 横寸 335mm となった。

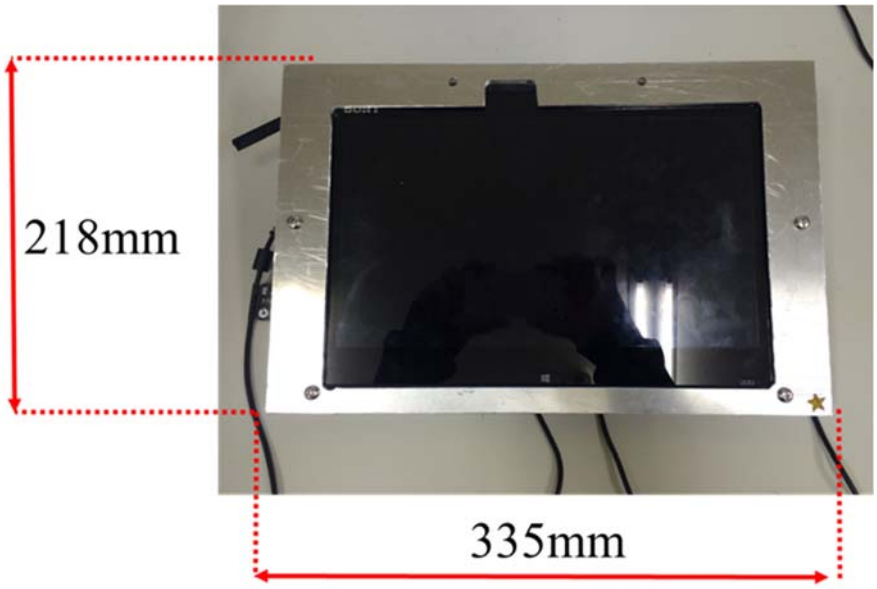

図 2-21　完成したカメラフレームの写真と寸法

2.5.4 考察・評価

完成したカメラフレームは 2.1.2 で述べた要求機能・制約条件を全て満たすことができた。

第3章 ステレオ法による距離計測 （橋本担当）

3.1 概要

3.1.1 目的

本章では、ステレオカメラを利用して対象物までの距離計測を行うプログラムについて解説する。家具シミュレーションを使用する上で、部屋の中における自己位置の推定を行う必要がある。そこで、ステレオ法を使用して、レンズと壁までの距離計測を行うことが可能なプログラムを作成する。

3.1.2 要求仕様

ステレオ法による距離計測を実現するための要求仕様を以下に示す。

1. ステレオ法を使用するために、カメラ 2 つの最適なレンズ間距離を求めること(図 3-1)
2. ステレオ法を使用して、レンズと被写体までの距離を算出すること

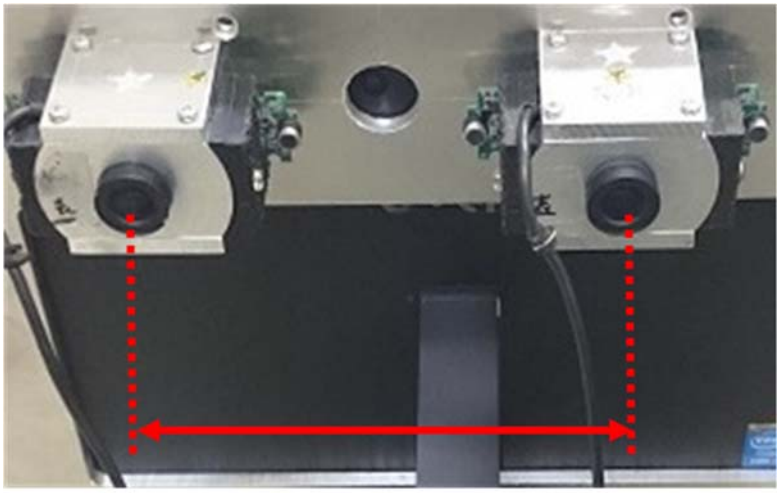

図 3-1　レンズ間の距離

3.2 使用したソフトウェア

本項では、距離計測を行うための開発環境を解説する。

3.2.1 Visual Studio 2015

プログラム作成の開発ツールとして、Visual Studio 2015(以降、VS 2015 と表記する)を使用する。VS 2015 は Microsoft 社が提供する開発ツールである。使用可能言語は C/C++、C#、JavaScript などが挙げられる。

3.2.2 Open CV

OpenCV とは、Intel 社が開発した画像処理・画像解析を行うライブラリである。言語は、C/C++、Java、Python が使用可能である。また、Windows、Linux、Mac OS、Android、iOS が公式サポートされている。

3.2.3 Open CV Sharp

OpenCVSharp とは、Schima 氏により作成された C#向けの OpenCV ラッパーライブラリである。今回の家具シミュレーションは C#で開発するので、OpenCVSharp を使用する。

3.3 レンズ間距離の実験

3.3.1 概要

本項では、ステレオ法を使用するのに最適な 2 つのカメラのレンズ間距離を調べる実験の方法と結果を解説する。

3.3.2 原理・理論

ステレオ法を使用するのに、2 つのカメラのレンズ間距離を設定する必要がある。レンズ間距離が大きすぎると、2 つのカメラから撮影した画像が、大きく異なる画像が撮影されステレオ法を使うことが不可能となる。反対に、レンズ間距離が小さすぎると、撮影した 2 つの画像に視差が生じなくなるため、やはりステレオ法を使うことはできない。よって、家具シミュレーションを行うのに最適なレンズ間距離を求める必要がある。

第 1 章で述べたステレオ法の式、

$$z=\frac{fd}{|x_1-x_2|}$$

にカメラと被写体までの距離 z(以降、z と表記する)、画面上の物体の重心$|x_1-x_2|$(以降、⊿p と表記する)を代入する。焦点距離 f は定数なので、レンズ間距離 d(以降、d と表記する)を変化させた時の⊿p と z の関係を調べる

3.3.3 重要なソースコード

物体の距離を認識するために、⊿p を求める必要がある。⊿p を求める方法として、3.4 でも行うように、物体の色を抽出してその重心を求めるという手法を利用する。

本実験では、図 3-2 のような赤い人形を使用する。左右のカメラで図 3-2 の人形を撮影し、人形の色である赤色の重臣を求めるプログラムを利用し、⊿p を求める。なお、以下のプログラムは参考文献の[3-1]を参考に作成した。

図 3-2　実験で使用する人形

まず、OpenCvSharp の関数である Cv.RGB を用いて、抽出する色の上限値、下限値を CvScalar として設定する。

```
CvScalar hsv_max = Cv.RGB(100, 255, 255);   // 抽出する HSV 色領域の上限
CvScalar hsv_min = Cv.RGB(0, 100, 150);     // 抽出する HSV 色領域の下限
```

CvScalar は double 型の値を 4 個まで格納することができる。また、Cv.RGB(x,y,z)の括弧内の数値を変化させることで、色要素を指定できる。なお、OpenCVSharp の画像データは RGB 表記する際、青色、緑色、赤色の順に設定する。つまり、Cv.RGB(x=青色、y=緑色、z=赤色)と数値を入力する。

```
float z = 150f;    //レンズと被写体の距離 z[cm]
float d = 12.5f;   //レンズ間距離 d[cm]
```

また、z、d の値を入力する必要があるので、変数を設定する。上式の場合、z は 150[cm]、d は 12.5[cm]となる。この値は、実験中に z、d の値を変えるときに変更する。

```
Cv.InRangeS(hsv1, hsv_min, hsv_max, mask1);          // 指定した範囲内の色抽出(二値化)
Cv.InRangeS(hsv2, hsv_min, hsv_max, mask2);          // 指定した範囲内の色抽出(二値化)

CvPoint pt1 = maxBlob1.Centroid;     // 面積が最大のラベルの重心座標を取得
CvPoint pt2 = maxBlob2.Centroid;     // 面積が最大のラベルの重心座標を取得
```

Cv.InRangeS は、入力配列の要素値が、指定した範囲内にある要素を出力する。これを利用し、二値化処理を行う。

CvPoint pt1(2) = maxBlob1.Centroid で、出力した重心の位置を取得できる。

3.3.4 実験方法

図 3-3 のように、ステレオカメラと被写体との距離 z を 100[cm]から 200[cm]まで 5[cm]刻みに変化させながら、⊿p を求めるプログラムを実行する。d を 12.5、17.5[cm]と設定し 5 回ずつ測定する。そして、z と⊿p のグラフを作成した後、最適な d の値を考察する。

図 3-3 レンズ間実験の様子

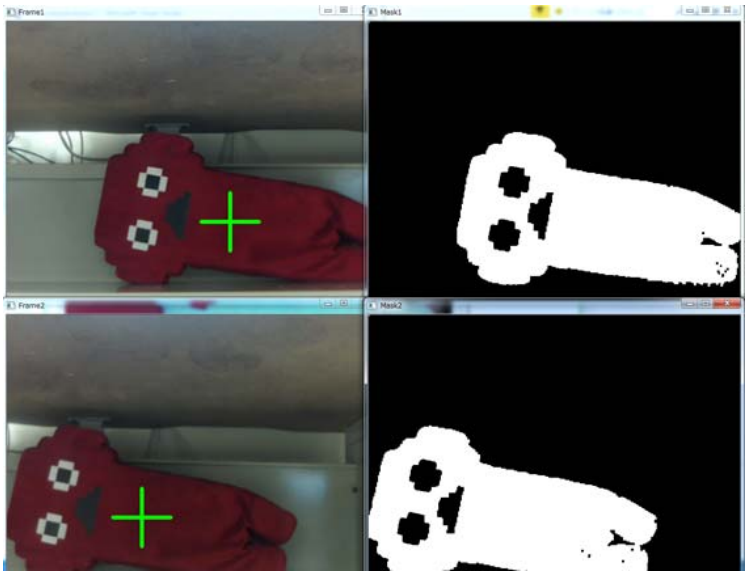

図 3-4　色抽出プログラム実行時の画像

図 3-4 に示されているように、画面上の緑の十字線の位置が抽出した色の重心であり、その x 座標の差が⊿p である。

3.3.5 実験結果

d=12.5、17.5[cm]と設定し実験を行った。d=12.5、17.5[cm]と設定した時の z と⊿p の関係を表 3-1 と表 3-2 に示した。

表 3-1　d=12.5[cm]の時の⊿p の値

距離z[cm]	平均値	1回目	2回目	3回目	4回目	5回目	標準偏差
100	166	177	160	155	150	182	12.51239
105	145	140	155	140	142	148	5.796551
110	119	100	105	116	109	111	5.418487
115	118	110	112	117	122	129	6.899275
120	92	90	95	97	87	93	3.555278
125	80	88	85	71	72	84	7.071068
130	69	77	71	64	66	71	4.534314
135	65	66	65	71	64	60	3.544009
140	52	55	50	48	47	60	4.857983
145	51	55	55	60	48	37	7.974961
150	38	30	44	40	35	41	4.939636
155	38	39	30	30	22	69	16.40732
160	36	28	26	45	44	37	7.874008
165	35	27	27	26	37	58	12.18195
170	34	28	27	40	34	41	5.830952
175	33	30	42	33	29	31	4.690416
180	33	31	33	29	28	44	5.761944
185	35	33	26	39	33	44	6.09918
190	32	22	38	33	33	34	5.329165
195	34	30	39	27	39	35	4.816638
200	30	25	24	36	38	27	5.830952

表 3-2　d=17.5[cm]の時の⊿p の値

距離z[cm]	平均値	1回目	2回目	3回目	4回目	5回目	標準偏差
100	358	312	322	390	370	396	34.71023
105	330	290	332	356	310	362	27.21764
110	320	280	352	290	360	318	32.02499
115	316	276	355	333	285	331	30.31831
120	304	268	322	311	300	319	19.54482
125	270	240	311	277	260	262	23.63895
130	249	221	270	199	245	310	38.63159
135	233	199	270	244	212	240	25.04396
140	204	179	186	250	227	178	29.22328
145	198	170	230	180	210	200	21.35416
150	180	155	159	230	222	134	38.59015
155	164	140	184	162	163	171	14.3527
160	141	129	111	170	146	149	19.86957
165	135	160	122	106	99	188	33.88215
170	135	155	100	97	162	161	29.91321
175	130	105	110	152	141	142	18.83614
180	128	111	125	149	142	113	15.23155
185	126	120	133	140	110	127	10.37304
190	125	115	130	135	110	135	10.48809
195	124	130	116	120	129	125	5.329165
200	124	102	124	139	122	133	12.60159

また、z と⊿p のグラフを表示したのが図 3-5 である。

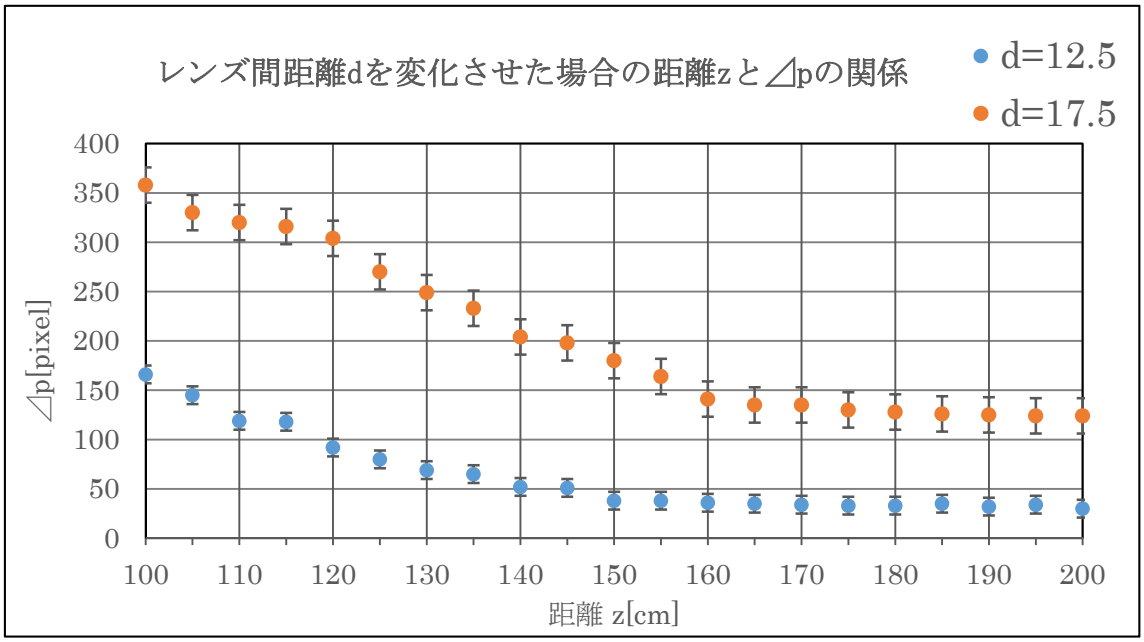

図 3-5 レンズ間距離 d を変化させた場合の距離 z と⊿p の関係

3.3.6 考察

表 3-1、図 3-5 より、d=12.5[cm]の場合、z が 100[cm]から 150[cm]の時、⊿p の変化を読み取ることが可能である。しかし、z≧150[cm]の場合は⊿p の値がほとんど変化していないことが分かる。d=17.5[cm]の場合、z=100[cm]から 160[cm]の時、⊿p の変化を読み取ることが可能であり、z≧160[cm]の場合、⊿p の値がほとんど変化していない。⊿p の変化を読み取れる z の範囲が広い d＝17.5 がステレオ法を利用する事に適していると考えられるが、⊿p の値が大きい数値を使用することになる。⊿p の値が大きい場合、1.1.4 節で述べた通り、左右のカメラで撮影した画像が大きく異なるため、立体を認識することが難しくなる。よって、⊿p を小さな範囲で利用できる d=12.5[cm]が立体を認識しやすいと考える。

3.4 距離計測法Ⅰ　色抽出法

3.4.1 概要

被写体の色情報を抽出し、抽出した色の重心位置を求め、距離を算出する。この方法を色抽出法とする。

3.4.2 原理・理論

距離計測を行う際、被写体に最適な色の抽出が必要となる。そこで、どのような色でも色抽出が可能となるようにプログラムを作成する。

抽出した色情報を用いて二値化処理を行い、距離を算出する。

3.4.3 ソースコード

3.3 のレンズ間実験でも使用したプログラムを改定する。3.3 では抽出する色を設定したが、今回のプログラムは画面上のクリックした箇所の色を抽出するように変更する。

本プログラムでは HSV を用いる。HSV とは、色相、輝度、彩度で色を表現する方法である。赤色、緑色、青色で色を表現する RGB では被写体以外の物体も抽出してしまうことが多いためである。

```
Cv.CvtColor(im1, hsv1, ColorConversion.BgrToHsv); // RGB色空間からHSV色空間に変換1
Cv.CvtColor(im2, hsv2, ColorConversion.BgrToHsv); // RGB 色空間から HSV 色空間に変換 2
```

このように、関数 Cv.CvtColor を用いると色空間の変換を行うことができる。RGB から HSV に変換する以外にも、グレースケール画像を作成することも可能である。

```
CvScalar hsv_max = Cv.RGB(360, 255, 255);   // HSV 色領域の上限
CvScalar hsv_min = Cv.RGB(point1.B - 50, point1.G - 50, point1.R - 50);      //
HSV 色領域の下限
```

上限値、下限値を Cv.Scalar に設定する。下限値の変数 point1.B(G、R)はクリックした点の色情報が格納されている。

```
CvWindow wImage = new CvWindow("無題", WindowMode.AutoSize, im1);
wImage.OnMouseCallback += delegate (MouseEvent ev, int x, int y, MouseEvent flags)
            {
             if (ev == MouseEvent.LButtonUp || (flags & MouseEvent.FlagLButton) == 0)
                {
                    prevPt = new CvPoint(-1, -1);
                }
                else if (ev == MouseEvent.LButtonDown)
                {
                    prevPt = new CvPoint(x, y);
                    c = prevPt.X;
                    cc = prevPt.Y;
                    Console.WriteLine(prevPt);
                }
              }
```

MouseEvent.LButtonDown によって、マウスの左ボタンでクリックした時の⊿p の表示を可能にしている。

3.4.4 動作確認テスト

図　3-6 に示した青色のドアのみを色抽出する実験を行った。抽出した後に 2 値化処理を行い、重心の位置が表示できるかを確認した。

図 3-6　抽出するドア

3.4.5 動作確認結果

図 3-7　動作確認テスト

図　3-7 が動作確認テストの結果である。左の画像が抽出前のドアの画像であり、青いドアを色抽出できていないことがわかる。そして、右の画像が画面上のドアをクリックし、色抽出を行った画像である。

3.4.6 考察

動作確認結果より、クリックした被写体の色抽出を行うことには成功したが、ドア以外の要素も抽出してしまっている。これは、ドア以外の色情報がクリックイベントにより設定した上限値、下限値に含まれているからであると考えられる。被写体以外の物体を撮影する画像に写さないことが対策として考えられる。また、具体的な距離計測までは至らなかった。

3.5 距離計測法Ⅱ　視差画像法

3.5.1 概要

2 枚の画像から視差画像を作成し、距離を計測する。この方法を視差画像法とする。この方法をステレオ法の 2 つ目の手法として動作確認を行う。

3.5.2 原理・理論

2 枚の画像を撮影し、2 枚の画像の視差を利用して合成した画像を作成する(図 3-8)。左の 2 枚の写真をもとに、右の視差画像を作成している。右の画像の色が白になるほど距離が近く、黒になるにつれて遠いことを表している。

図 3-8　2 枚の画像から視差画像製作

3.5.3 ソースコード

視差画像を作成するのに、Cv.FindStereoCorrespondenceBM を用いる。なお、以下のプログラムは参考文献の[3-1]を参考に作成した。

```
Cv.FindStereoCorrespondenceBM(imgLeft, imgRight, dispBM, stateBM);
```

括弧内の imgLeft、imgRight はそれぞれ左右の画像である。dispBM、stateBM は左の視差画像、右の視差画像をそれぞれ作成する。

そして、配列に対して線形変換を行う関数 Cv.ConvertScale を用いて、左右それぞれの視差画像を合成し、一枚の視差画像を作成する。

```
Cv.ConvertScale(dst, dst, 12);
```

3.5.4 動作確認テスト

図 3-9 はステレオカメラを利用して撮影した画像である。図 3-9 の左の画像がステレオカメラの左のレンズ、右の画像がステレオカメラの右のレンズで撮影した画像である。レンズから近い順に青いイス、画像左の肉の人形、赤い人形を配置し、視差画像を作り、距離を算出することができるか検証する。

図 3-9　撮影した画像

3.5.5 動作確認結果

図 3-10　得られた視差画像

実際にプログラムを実行して得られた視差画像が図 3-10 である。画像中央のカメラから一番近い青いイスが視差画像では白色になっていることがわかる。しかし、肉の人形、赤い人形などの被写体は距離が遠くても黒色にならなかった。

3.5.6 考察

3.3 節での実験より、z が 150[cm]以上であると⊿p の範囲を読み取ることはできなくなるため、距離計測はできなくなる事が分かった。図 3-9 のカメラとイスの z は 105[cm]、パイプイス、肉の人形は 155[cm]、赤い人形は 600[cm]である。z が 150[cm]以上であるため、パイプイス、肉の人形、赤い人形は距離計測ができなくなっていると考えられる。

3.6 2 つの方法の比較

本章では、色抽出法と視差画像法の 2 つの方法で距離計測を試みた。色抽出法は、輝度と彩度に大きく依存していることが特徴として挙げられる。被写体が照明の光等により輝度に違いが生じると、設定した色の上限値もしくは下限値の範囲外となり、色抽出ができなくなる。

視差画像法は、輝度と彩度に関係なく距離を把握することができる。しかし、視差画像を作成するために、プログラム中の変数を変化させなければならない。適切な変数を発見することはできなかった。

また、どちらの方法でも被写体と 2 つのカメラの角度を調整し、ステレオ法を使用するのに最適な画像を撮影する事が必須である。

3.7 距離計測の最終的評価

3.7.1 要求仕様との比較

3.1.2 で記述したように、距離計測を実現するため、以下の要求仕様を満たす必要があった。

1. ステレオ法を使用するために、カメラ 2 つの最適なレンズ間距離を求めること
2. ステレオ法を用いて、レンズと被写体までの距離を求めること

まず 1.について、レンズ間実験を行い、d=12.5[cm]、17.5[cm]を比較すると、d=17.5[cm]の時は使用可能な z の範囲が広くなる。しかし、⊿p の範囲が小さい d=12.5[cm]が立体視を行うのに適していると考えられる。次に 2.について、色抽出法と視差画像法の 2 つの方法で研究を行った。色抽出法では、被写体の色の抽出を行うことができた。また、視差画像法は画像内の一部のおおまかな距離を把握することができた。しかし、どちらの方法でも具体的な距離を計測することはできなかった。

3.7.2 考察

最終的な評価として、ステレオ法を使用するのに、d=12.5[cm]の場合、z=150[cm]までの距離ならば、距離を計測する事ができると予想ができた。また、被写体の色の抽出を行うこと、視差画像によって大まかな距離判別を行うこともできたと言える。しかし、具体的な距離計測を行うプログラムを作成することはできなかった。また、色抽出法と視差画像法のどちらが家具シミュレーションに適しているか、判断もできていない。プログラム内の変数や関数を調整し、距離計測ができるプログラムに改良していく必要があると言える。

第4章 シミュレーション用ソフトウェアの開発

（片山担当）

4.1 概要

4.1.1 目的

本章ではシミュレーション用ソフトウェアの開発について解説する。本研究ではタブレット PC を持ち、自由に移動しながらシミュレーションを行うことを目的としている。家具配置のシミュレーション用の映像は、タブレット PC に内蔵されているカメラ(以下、内蔵カメラと称する)から取得したものを使用する。そのため、本ソフトウェアでは以下の 3 点が必要となる。

1. 内蔵カメラの映像の取得及び家具の CG との合成(図 4-1 左)
2. タブレット PC に内蔵されている加速度センサ及びジャイロセンサを利用した位置と方向の推定用プログラムの作成(図 4-1 右)
3. 開発したソフトウェアと 3 章で解説したステレオ法による距離計測の合致。

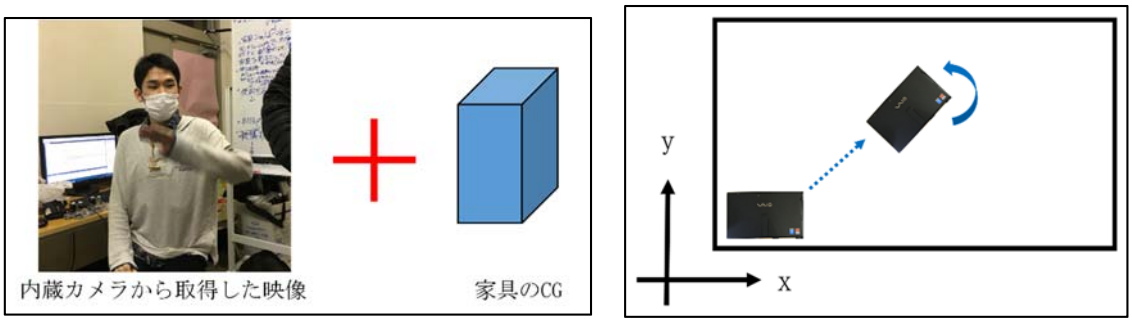

図 4-1 （左）内蔵カメラから取得した映像と家具の CG の合成について
（右）タブレット PC の位置と方向の推定について

4.1.2 要求仕様

シミュレーション用ソフトウェアを実現するため、以下の要求仕様を満たす必要がある。

1. 内蔵カメラから映像を取得しタブレット PC の画面上に出力できること
2. 取得した映像と家具の CG を合成すること
3. タブレット PC を持ち歩いた際に現在位置をリアルタイムで取得すること

4.2 使用したソフトウェア

4.2.1 Unity について

本研究では、Unity Technologies より開発されたマルチプラットフォーム対応の統合開発エンジンである Unity を使用する(図 4-2)。Unity が対応しているプラットフォームにおいて代表的なものは Mac や iOS、Android、そして本研究において使用する Windows である。

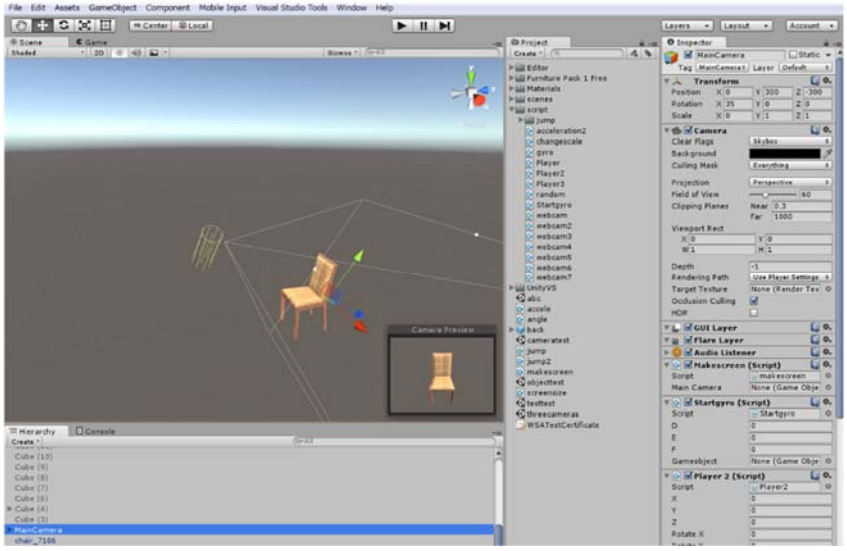

図 4-2　使用した Unity でソフトウェアを開発している画面［4-1］

4.2.2 Visual Studio for Unity

本研究では、Visual Studio 上で Unity 開発を行うため、Visual Studio for Unity というソフトウェアを使用する。このソフトウェアを使用することで、ソフトウェアを Visual Studio で開発でき、タブレット PC 用の形式である Windows ストアアプリとして出力することができる。

4.3 Windows Store App

Unity では従来の Windows ソフトウェアの他に Windows ストアアプリ形式のソフトウェアも出力することができる。Windows ストアアプリとは Windows ストアアプリ対応のソフトウェアのことで、Windows 8 で動作する新しいタイプのソフトウェアのことを指す。従来の Windows ソフトウェアでは、タブレット PC に内蔵されている加速度センサとジャイロセンサを使用することができないが、Windows ストアアプリならばそれらを使用することができる。本研究では、タブレット PC に内蔵されている加速度センサとジャイロセンサを使用する必要があるため、Windows ストアアプリを使用する。

4.4 開発したソフトウェアについて

4.4.1 概要

本研究では、内蔵カメラから映像を取得し、その映像に家具の CG を合成した後、画面上に出力するソフトウェアを開発する。屋内での利用を目的としているため、GPS では正確な自己位置の推定は期待できない。そこで GPS に代わり加速度センサ及びジャイロセンサからそれぞれ加速度、角度を取得し、それらを利用してタブレットの位置と方向をリアルタイムで推定するソフトウェアを開発する。

本研究の要求仕様を満たすソフトウェアは次のような構成となっている。まず、モデルルーム内での利用を想定しているため、壁までの距離を 3 章で解説したステレオ法による距離計測で求める。これに加え、ステレオカメラが向いている方向以外の動きを検知するため加速度センサを使用する。更に、ジャイロセンサを使用して、タブレット PC の向いている方向を取得する。この 3 つを組み合わせることで自己の位置と向きを推定する。

本研究において内蔵カメラを使用する理由として次のことが挙げられる。本研究では、3 章で解説したステレオ法による距離計測においても 2 台のカメラを使用する。しかし、これらのカメラは 2 台並べて配置するため、それぞれの映像が左右に偏ってしまう可能性がある。また、本研究開始時点ではステレオカメラの取り付け位置に関しても未定であった。以上の 2 つの理由から、画面上に出力した際にタブレット PC と映像との位置関係に違和感が生じることを避けるため内蔵カメラを使用することとなった。

4.4.2 動作原理

開発したソフトウェアは次のような構成となっている(図 4-3)。

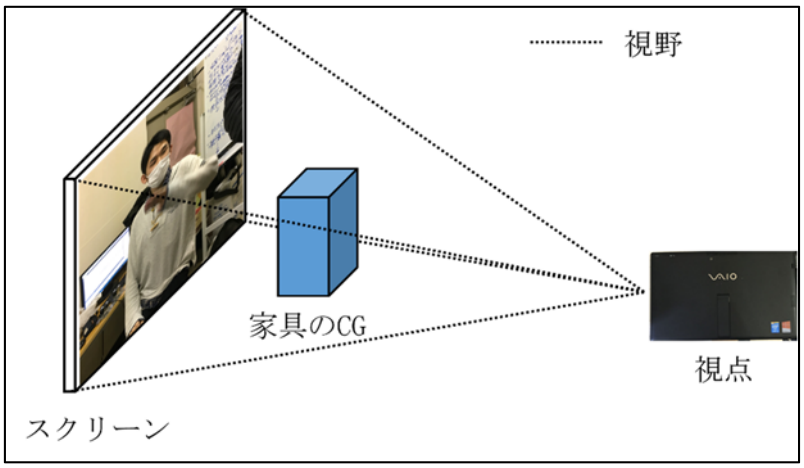

図 4-3　開発したソフトウェア内の視点、スクリーン、家具の CG の配置

まず、ソフトウェア内部には使用者が持つタブレット PC の位置や向きと連動した視点が配置されている。また、視点とは別に、距離を置いて内蔵カメラより取得した映像を出力するためのスクリーンが仮想空間内に配置されている。このスクリーンは距離に応じて、常に視点の視野全体を埋める大きさを維持している。

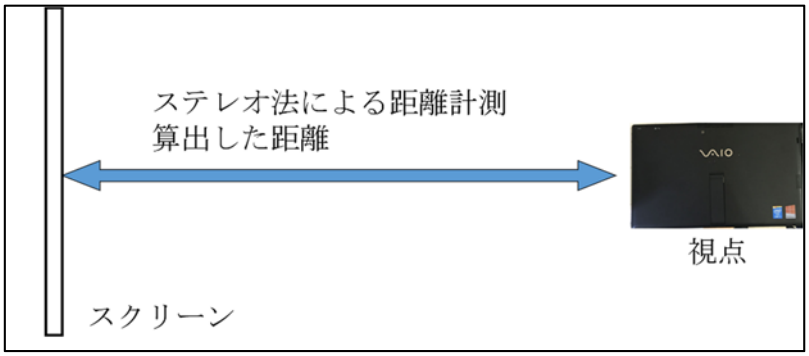

図 4-4　開発したソフトウェア内でステレオ法による距離計測を使用する点

また、視点とスクリーンとの距離にはステレオ法による距離計測より取得した壁までの距離を使用する(図　4-4)。この視点とスクリーンとの間に任意の位置に配置しておいた家具の CG が入り込むことで、内蔵カメラより取得した映像に家具の CG が映って見える(図 4-5)。このような構成にすることで壁と家具の CG、あるいは使用者と家具の CG との距離感を再現することができる。これは、本研究の対象がモデルルームであり、ステレオ法による距離計測を使用する対象が壁や天井、床などであり、間に遮るものがないことが前提となっている。

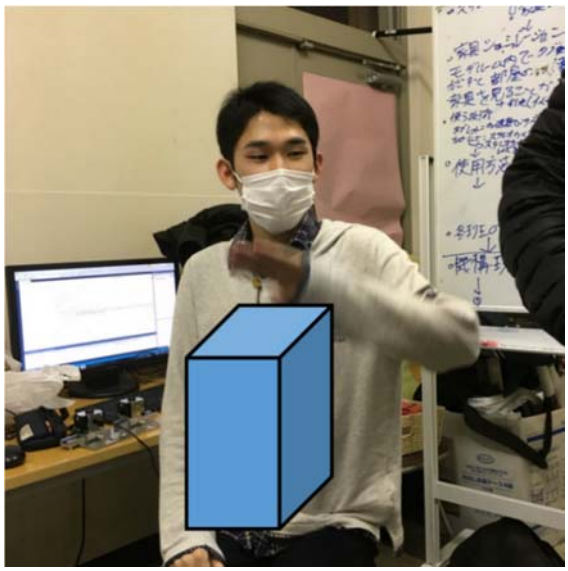

図 4-5　開発したソフトウェアが出力する映像の目標像

4.4.3 加速度センサ

本研究において、加速度センサは自己位置の推定の補助に使用する(図 4-6)。ステレオ法による距離計測では、ステレオカメラが向いた方向にある対象との距離を計測することができる。しかし、この手法だけではそれ以外の方向に対して自己位置の推定を行うことができない。そのため、加速度センサを使用することで、ステレオカメラの方向以外への移動をその加速度の積分から算出し、自己位置の推定の補助とする。

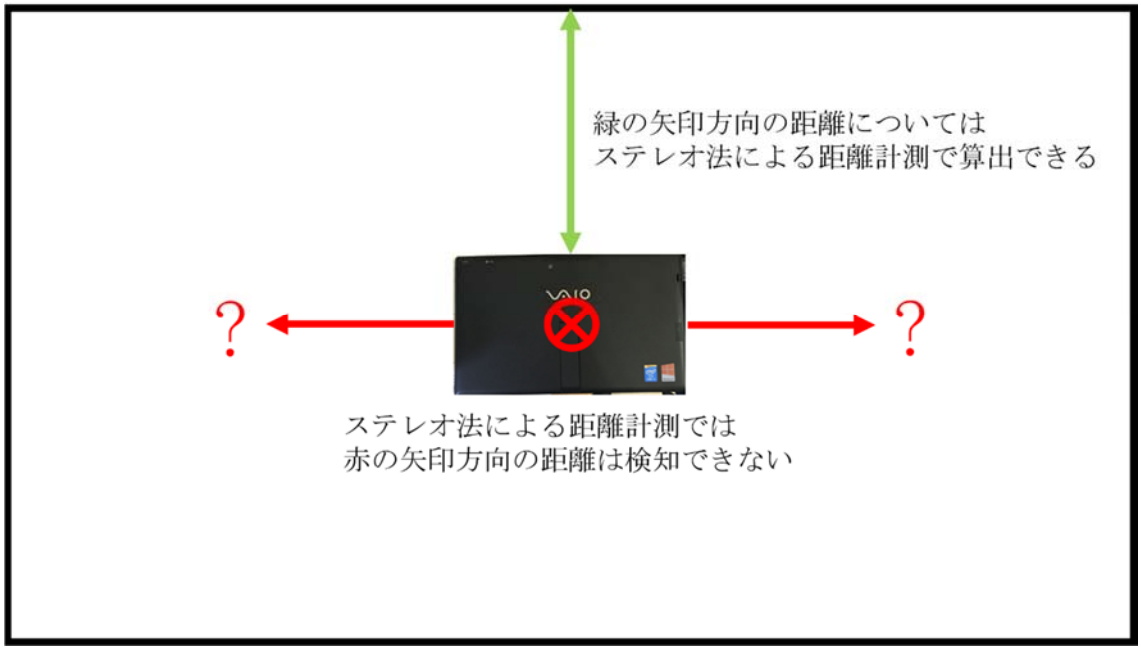

図 4-6　加速度センサの用途

4.4.4 ジャイロセンサ

4.4.3 で解説した方法は、タブレットが壁に対して正面を向いているときしか適用できない。例えば図　4-7 のようにタブレットが壁に対して斜めを向いているとき、計測されるのは緑の矢印の長さであり、求めたい壁までの距離とは異なる。そのため、ジャイロセンサを使用することでタブレット PC が向いている方向を検知することで、自己位置の推定の補助とする。

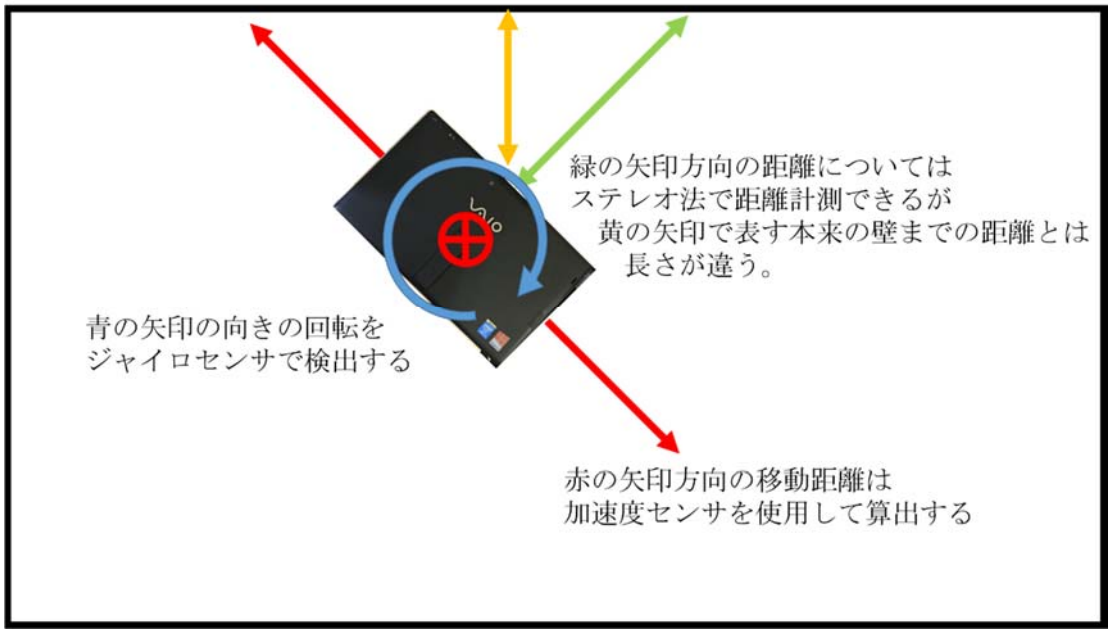

図 4-7　ジャイロセンサの用途

4.5 内蔵カメラの出力実験

4.5.1 概要

本節は、内蔵カメラから映像を取得し、家具の CG と合成した後、画面上に出力できるかを検証することを目的とした実験を行う。

4.5.2 動作原理

以下に、内蔵カメラの映像を取得し出力する実験に関連するソースコードとその解説を記載する。作成したプログラムは参考文献[4-1]を参考にした。

```
public int Width = 640;
public int Height = 360;
public int FPS = 60;
```

上記、3 行のソースコードによって、出力する映像の解像度及びフレームレートを設定している。1 行目の値を変更することでカメラの横の解像度、2 行目の値を変更することでカメラの縦の解像度をピクセル単位で変更することができる。また、3 行目の値を変更することで映像のフレームレートを変更することができる。

```
WebCamDevice[] devices = WebCamTexture.devices;
WebCamTexture webcamTexture = new WebCamTexture(devices[0].name, Width, Height,
FPS);
```

また、このソースコードを使用することでカメラの映像を出力することができる。2 行目の括弧内にある devices[]の中の数字を変更することで、その番号に対応したカメラの映像を出力する。

4.5.3 ソースコード

Windows ストアアプリ形式で出力したソフトウェアをタブレット PC で起動し、スクリーンに映像を出力できるかどうかを確認する(図 4-8)。また、その際に映像に家具の CG が表示されているかどうかも同時に確認する。

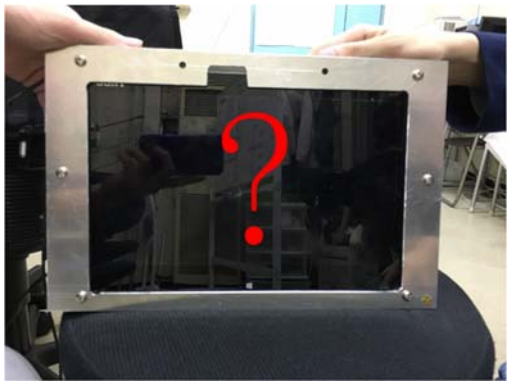

図 4-8　ソフトウェア起動前のタブレット PC

4.5.4 実験方法

映像の解像度が width = 640、height = 360 ピクセル以下のとき、映像の出力ができた。下の図 4-9 の左の画像が出力に失敗したもの、右の画像が成功したもの(右)である。

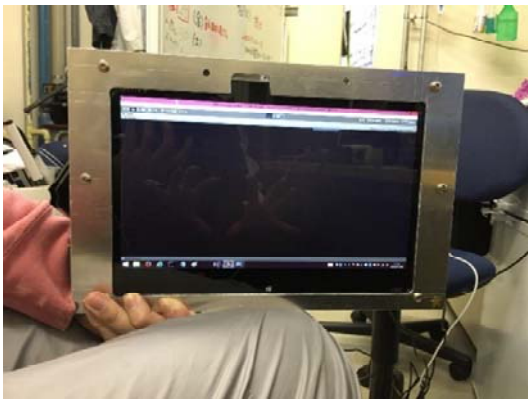

図 4-9　(左) 解像度が 640 × 360 より大きい時
(右) 解像度が 640 × 360 以下の時

また、映像に家具の CG を表示させることにも成功した。下の図 4-10 の左の画像が家具の CG を出力したもの、右の画像が実際の部屋のものである。図の赤い丸で囲ってある箇所にイスの CG が出力されているかどうかを比較している。

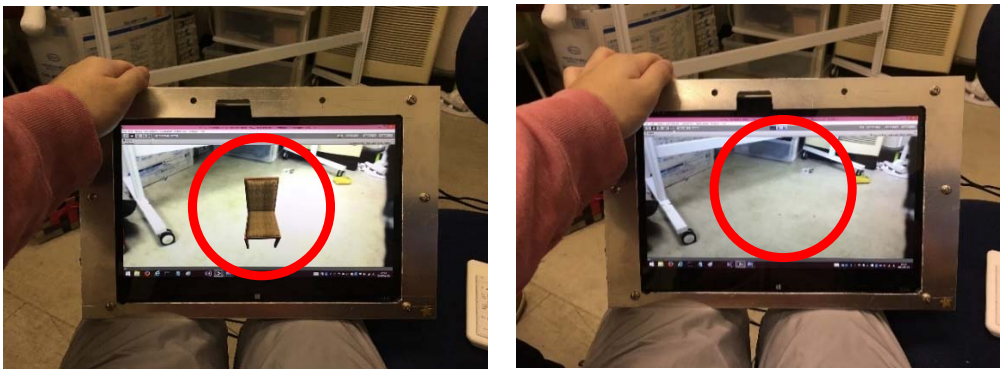

図 4-10 （左）家具の CG を出力したもの
（右）家具の CG を出力していないもの

4.5.5 考察

Unity で開発したソフトウェアは、タブレット PC に大きな負荷をかけることが分かった。内蔵カメラで取得した映像を出力すると、フレームレートを 60 に指定していたとしても映像がコマ送りのように見えてしまう。これはソフトウェアがタブレット PC にかけている負荷とタブレット PC の処理能力の問題であると考えられる。比較として、研究室の備品であるタブレット PC ではなく、班員の私物のタブレット PC を使用したところ、同じソースコードのソフトウェアであっても、映像が滑らかに出力されることを確認した。下の表 4-1 に研究室の備品のものと、私物のものを記す。

表 4-1 研究室の備品と私物のタブレット PC の比較

	研究室の備品	私物
プロセッサ	Intel(R) Core(TM) i3-4020Y CPU @1.50GHz	Intel(R) Core(TM) i5-3317U CPU @1.70GHz
実装メモリ(RAM)	4.00GB	8.00GB
オンボードグラフィックス	Intel(R) HD Graphics Family	Intel(R) HD Graphics 4000

本研究を実用化するにあたっては、人が映像の動きに違和感を持たなくなるといわれている 50～60 フレームレートまで、フレームレートを向上させ出力できる環境を作る必要がある。これはソフトウェアがタブレット PC にかける負荷を抑えることや、タブレット PC の高性能化で改善できると考えられる。

4.6 加速度センサの実験

4.6.1 概要

本節は、タブレット PC に内蔵されている加速度センサを使用し、タブレット PC の移動距離を算出する実験を行い、本研究に適しているかどうかを判断する。

4.6.2 ソースコード

以下に、タブレット PC が x 軸から受ける加速度を取得し、移動距離を算出するプログラムのソースコードと解説を記載する。y、z 軸についても同様の処理を行っている。作成したプログラムは参考文献[4-2]と[4-3]を参考に作成した。

```
xac = Input.gyro.userAcceleration.x;
```

Input.gyro.userAcceleration.x を使用することで、タブレット PC が受けている x 軸方向の加速度から重力加速度を除いた値を xac として取得することができる。ここで xac を 1 フレーム目の加速度として設定する。

```
xaca = Input.gyro.userAcceleration.x;
mx = xaca - xac;
```

次に xaca を 2 フレーム目に取得した加速度と定義し、xaca と xac の差を mx として定義する。この mx を使用し、手振れなどによって生じる微細な加速度を除去するため、次の条件をノイズフィルタとして設定した。

```
if (System.Math.Abs(mx) > 0.002f){
        intervalTimeX += time;
        if (0.1 >= intervalTimeX ){
                vx = 0;}
        else{
                vx = mx * intervalTimeX + fx;}}
else{
        vx = 0;
        intervalTimeX = 0;}
dx = vx * time;
```

mx の絶対値が 0.002 より大きい時、フレーム間の時間 time を intervalTimeX に加え、その値が 0.1 以下の時は速度 vx を 0、それ以外の場合は速度 vx を加速度の差 mx を時間について積分した値として定義する。また、mx の値が 0.002 未満だった場合は移動していないと定義し、速度 vx を 0 とする。移動した距離 dx は速度 vx に時間をかけて算出する。

```
fx = vx;
xac = xaca;
```

最後に、fx を前フレームで算出した速度、xac を前フレームで取得した加速度と置き換える。

4.6.3 実験方法

タブレット PC を直線状に 100[cm]動かし、加速度センサから取得した加速度を元に算出した移動距離と比較する。移動する時間を 1[s]から 3[s]まで 0.5[s]毎に増加させ、与える加速度の値を変え、算出した移動距離の誤差を記録する。

4.6.4 実験結果

図 4-11 の実験結果から、実際に移動させた距離と加速度センサを使用して算出した移動距離との誤差が分かる。グラフの縦軸は加速度センサを使用して算出した移動距離[cm]、横軸は移動速度[cm/s]としている。また、グラフ上に記載した赤い線が本実験で実際に移動させた 100[cm]、黒い線は 5 回実験した標準偏差を示している。

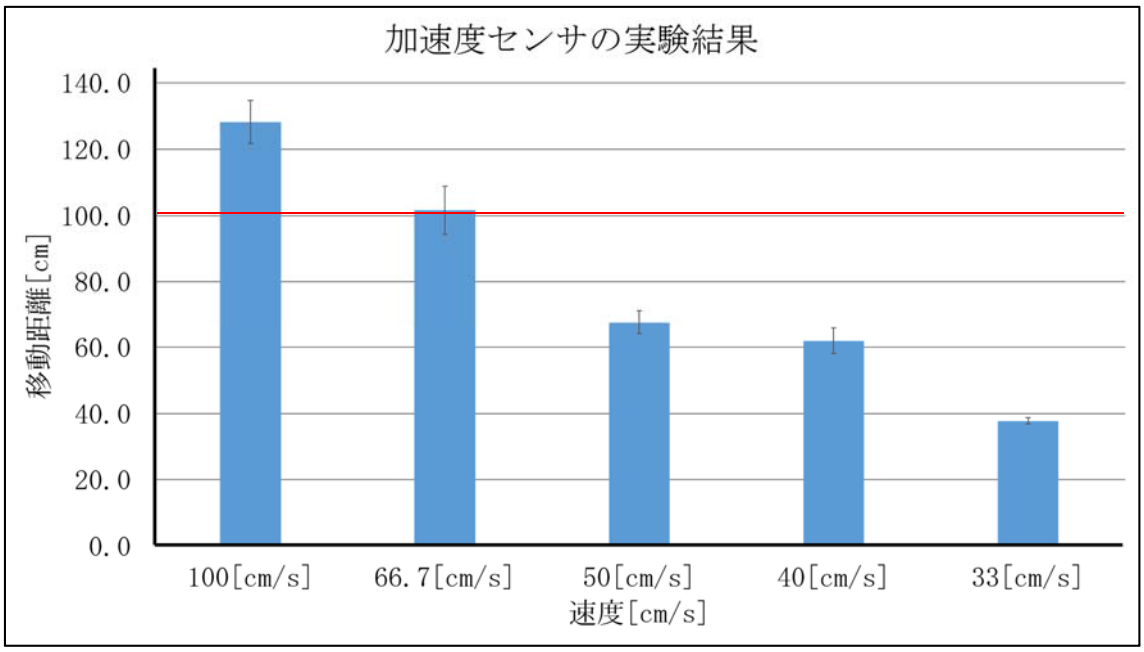

図 4-11　加速度センサの実験結果のグラフ

4.6.5 考察

移動距離を算出するにあたって、66.7[cm/s]の速度で移動させたものが実際に移動させた距離との差が最も小さいという結果になった。与えた速度が大きすぎた場合、初動時と停止時に大きな慣性の影響を受け、誤差が生じてしまう。逆に小さいと値と手振れによるノイズが相対的に大きくなり、値の有用性を判断することが難しくなる。そのため、プログラムによるノイズフィルタによって与えられた加速度を加速度と認識することができなくなる。このことが実際の移動距離と算出した移動距離の差を大きくしてしまう要因の 1 つであると考えられる。また、加速度センサが本研究に適しているかどうかについてだが、使用者が家具の配置を見て歩く際、大きく横に動いてしまうと、この実験結果からでは自己位置の推定は困難であると考えられる。そのため、今回利用したアルゴリズムは、本研究で使用する距離計測の目的に適していないと判断する。

4.7 ジャイロセンサの実験

4.7.1 概要

本節では、ジャイロセンサを使用し、タブレットPCが向いている方向を計測する実験を行い、本研究の目的に適しているかどうかを判断する。

また、ジャイロセンサの動作確認を行う際、現在向いている角度を初期状態として0[度]とするのではなく、ソフトウェア内で指定した初期の角度との差として取得することができることがわかった。このことから、本実験で使用したジャイロセンサは角加速度や角速度の値から積分し角度を算出するのではなく、リアルタイムに角度そのものを検出することができるものであると考えられる。これについても同時に検証を行う。

4.7.2 ソースコード

以下に、ジャイロセンサを使用して角度を取得する際のソースコードと解説を記載する。作成したプログラムは参考文献[4-4]を参考にした。

```
Input.gyro.enabled = true;
if (Input.gyro.enabled) {
        ini_gyro = Input.gyro.attitude;
this.transform.localRotation = Quaternion.Euler(90, 0, 0) * (new Quaternion(-
ini_gyro.x, -ini_gyro.y, ini_gyro.z, ini_gyro.w));
        }
```

ジャイロセンサを使用し、その値を四元数Quaternionからオイラー角へと変換するプログラムである。Input.gyro.enabled = true;によって、ジャイロセンサを使用することを宣言し、ini_gyro.attitude を使用することでジャイロセンサの値を使用することができる。その後、this.transform.localRotation によって視点の角度を変更させることができる。変更する値は(new Quaternion(-ini_gyro.x, -ini_gyro.y, ini_gyro.z, ini_gyro.w));によってジャイロセンサの値を読み取り、Quaternion.Euler(90, 0, 0)によってオイラー角に変更したものを適用している。

4.7.3 実験方法

タブレット PC を x、y、z 軸まわりにそれぞれ 90、180[度]回転させ、ジャイロセンサから取得した角度と比較する。回転する時間を 1[s]から 5[s]まで 1 秒毎に増加し、与える角加速度の値を変え、取得した角度を小数点第 1 位で四捨五入した後、誤差を記録する。

4.7.4 実験結果

実験結果から、実際に回転させた角度とジャイロセンサを使用して検出した角度との誤差が分かる(図 4-12)。グラフの縦軸はジャイロセンサを使用して検出した角度[度]、横軸は x、y、z 軸のいずれかと 90、180[度]回転を組み合わせたものとしている。また、グラフ上に記載した黒い線が本実験で実際に回転させた 90、180[度]を、各棒の上部にある黒い線は 5 回実験した標準偏差を示している。青、緑、黄、橙、赤のグラフの色分けについては回転させた際の所要時間を示し、それぞれ 1、2、3、4、5[s]としている。回転させる速さに依存せず、一定の角度を検出できていることがわかる。ただし、90[度]回転させたときに 94[度]と読み取られるなど、結果に誤差が乗ることも分かった。

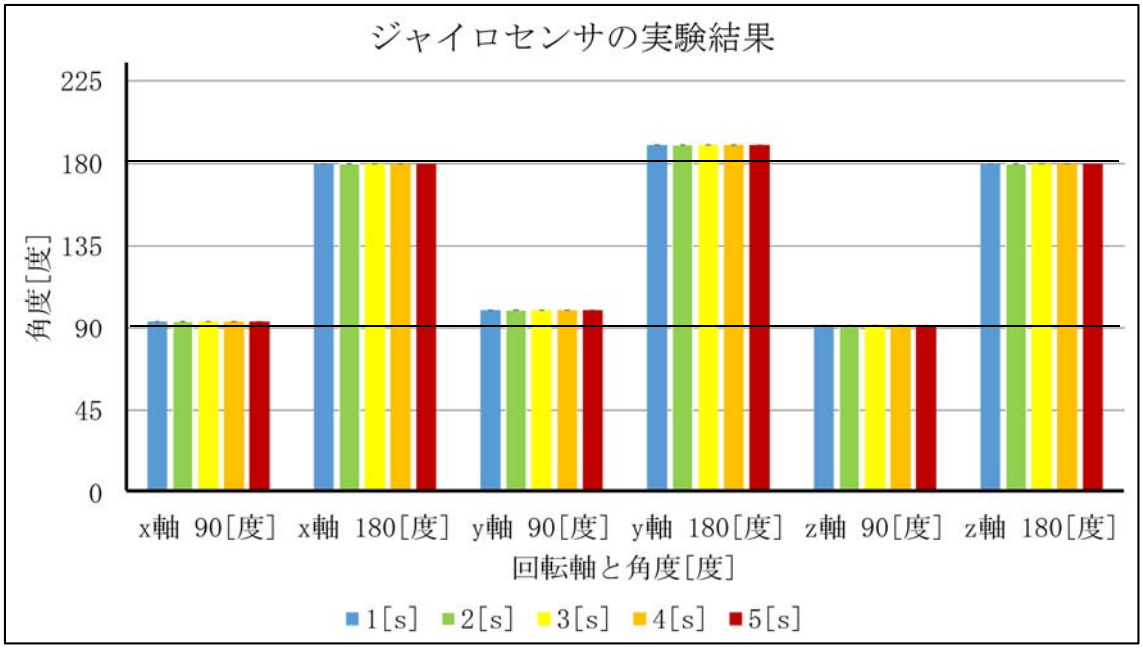

図 4-12　ジャイロセンサの実験結果のグラフ

4.7.5 考察

本実験で使用したジャイロセンサは、四元数を取得し角度を検出することができるものである。そのため、角加速度や角速度を積分する必要があるジャイロセンサとは違い、積分による角度のずれを蓄積することがない。これは、回転させる時間によらず、検出された角度が一定であるという実験結果からも判断できる。このことから、本実験で使用したジャイロセンサは本研究で実現しようとしている家具シミュレーションに非常に適していると判断できる。

また、実験結果について、なぜ90[度]回転させたときに94[度]と検出されるなどの誤差が生じているのかについて考察した際、内蔵カメラの出力実験の考察でも行ったように研究室の備品であるタブレットPCではなく、班員の私物のタブレットPCを使用したところ、同じソースコードであっても角度の誤差はほぼ 0[度]といってよい結果を得ることができた。このことから、タブレットPCの性能や搭載されているジャイロセンサの性能によって角度を検出する際の精度にずれが生じているのではないかと考えられる。

4.8 OpenCV for Unity について

4.8.1 概要

ステレオ法による距離計測のために 3 章で使用しているプログラムを Unity 内で使用するため、OpenCV を Unity 内で使用できる環境を作らなければならない。そこで、OpenCV for Unity という$95 で購入できる有償のプラグインについて検討した。このプラグインを購入することで、Unity 内でソフトウェアの開発に OpenCV を使用することができ、ステレオ法による距離計測を Unity で開発したソフトウェアと融合させられると考えた。そのためには、まず Unity でステレオカメラに使用するカメラ 2 台と、ソフトウェアで使用する内蔵カメラ 1 台の合計 3 台のカメラを同時に使用できるか検証する必要がある。

4.8.2 3 つのカメラの映像出力

前述した通り、3 台のカメラを Unity で同時に使用できるかを検証するため、3 台のカメラの映像を同時に取得し、画面上に 3 つ同時に出力するソフトウェアを開発した。このソフトウェアを使用し、3 台のカメラを同時に使用できることは検証できた(図 4-13)。しかし、卒論提出の期限まで時間が残り少ないことから OpenCV for Unity の購入は見送る結果となった。

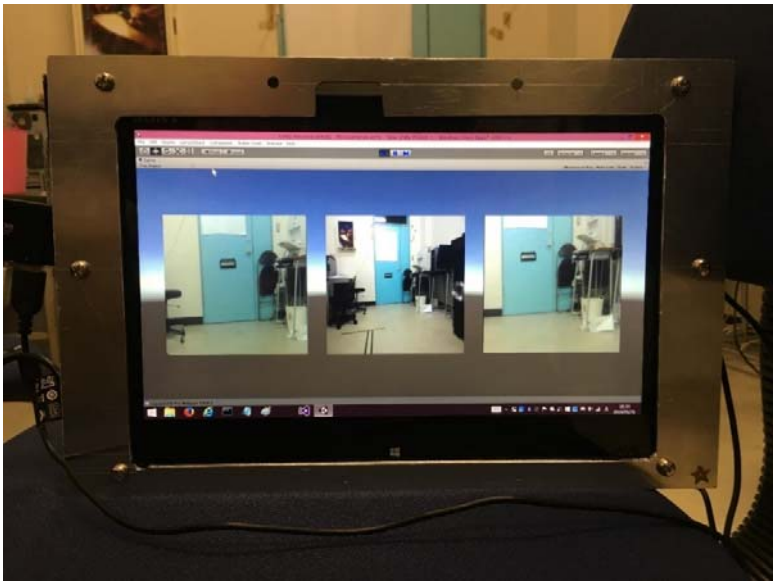

図 4-13　3 つのカメラの映像出力実験の結果

4.9 開発したソフトウェアの最終的評価

4.9.1 要求仕様との比較

4.1.2 に記述したように、シミュレーション用ソフトウェアを実現するため、以下の要求仕様を満たす必要がある。

1. 内蔵カメラから映像を取得しタブレット PC の画面上に出力できること
2. 取得した映像と家具の CG を合成すること
3. タブレット PC を持ち歩いた際に現在位置をリアルタイムで取得すること
4. 家具の配置を設定する画面の作成

まず 1.について、解像度は下がるものの映像を出力することはできた。ただし、タブレット PC の性能によって、映像の滑らかさに違いが生じるため、本研究で開発したソフトウェアを使用する際はタブレット PC の性能に注意する必要がある。次に 2.について、取得した映像に家具の CG を合成し､映像として画面上に出力することもできた｡更に 3.について、タブレット PC を持ち歩いた際、加速度センサから得られる位置情報は誤差が大きく、現状のプログラムでは位置を知ることができているとは言えない状態である｡また､ステレオ法による距離計測に関しても統合させることができていないため、3.については要求を満たしていないと判断した。最後に 4.について、本研究において重要となる家具の配置についてである。当初の予定では家具の配置を設定する画面を作成する予定だった。しかし、時間の都合上、これの作成に着手することができなかった。

4.9.2 考察

最終的な評価として､本章で 3 つの実験を踏まえて考察すると､まだ至らない点は数多く存在しているものの、タブレット PC を使用した家具シミュレーションシステムの開発は実用化できるものであると考えられる。特にタブレット PC に内蔵されているジャイロセンサに関しては､本研究を実用化するにあたって適していると考えられる。また､加速度センサについては､実験に使用したアルゴリズムに問題があり､実験の段階で誤差が大きくなってしまった。そのため、本研究に適しているかは今回の実験結果からは判断できない。

第5章 家具シミュレーションシステムの動作確認

（鈴木担当）

5.1 概要

タブレットPCに本研究で製作したカメラフレームを取り付けて、加速度センサ、ジャイロセンサを使用した家具シミュレーションを行う。ステレオカメラは、距離測量のプログラムをシステムの一部として組み込むことができていないため、本実験では使用しない。実験は工学院大学犬目キャンパスの研究室内で行った(図 5-1)。

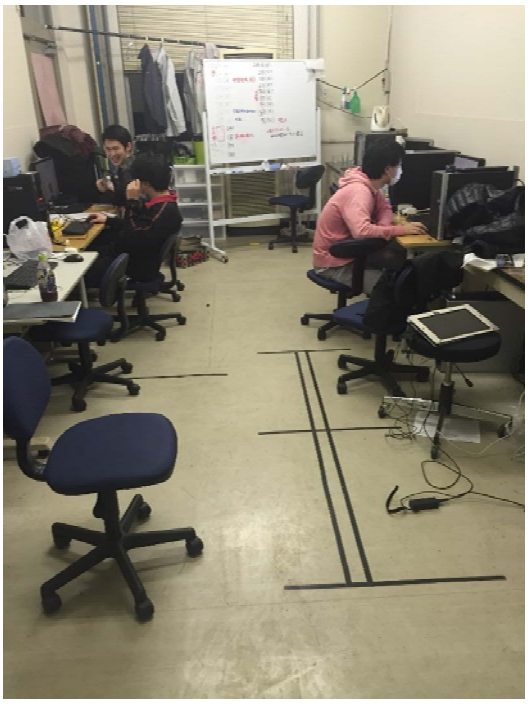

図 5-1　実験を行う研究室

5.2 実験方法

本実験は、A 点を初期位置として、B 点、C 点、D 点の順に 1m ずつ移動した時、システム内の家具の見え方がどのように変化しているかを検証するものである(図 5-2)。システム内で A 点と B 点と C 点と D 点にそれぞれ家具が配置されている。図 5-3 は、システム内に配置されている家具を研究室の背景に合成した図である。A 点と C 点には椅子、B 点と D 点には机が配置されている(図 5-4)。実験する際の視点方向は、A 点の時に B 点の方向、B 点の時に C 点の方向、C 点の時に D 点の方向、D 点の時に A 点の方向を向く。点から点に移動する際の速度は 100cm/s を目安として移動する。また、D 点から A 点に戻ったとき、家具の見え方も元通りになっているかどうかも検証する。

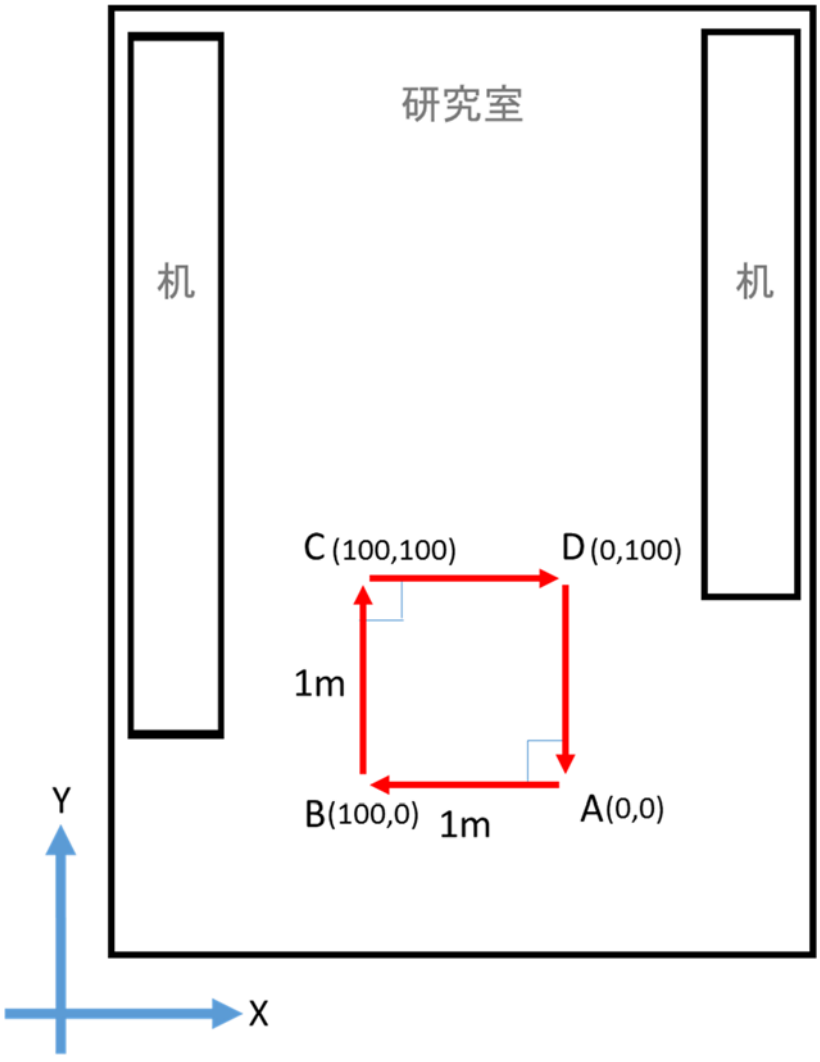

図 5-2　研究室内の移動手順

図 5-3　研究室の背景に CG の椅子と机が合成されている図

図 5-4 (左)A 点と C 点に配置されている椅子の CG
(右)B 点と D 点に配置されている机の CG

5.3 実験結果

実験の結果、自己位置を変更すると、家具の見え方も変化することが確認できた(図 5-5)。図 5-5 の赤丸の部分が家具の位置を示している。A 点では B 点に配置されている家具、B 点では C 点に配置されている家具の認識に成功している。しかし、C 点では家具を認識することはできているものの、写っている家具が D 点と A 点の家具であるならば、D 点の机が左側、A 点の椅子が右側に写ることが理想だが、写り方が逆になってしまっている。D 点でも C 点と同様に家具の認識には成功しているが、A 点に配置されている椅子ではなく、机が写ってしまっている。

また、D 点まで移動したあとに A 点に戻ったとき、家具の位置も最初に A 点で観測した位置に戻っていることが理想であったが、図 5-5 のように最初に A 点で観測した家具の場所と、最後に A 点で観測した家具の場所がずれてしまうことが確認できた。

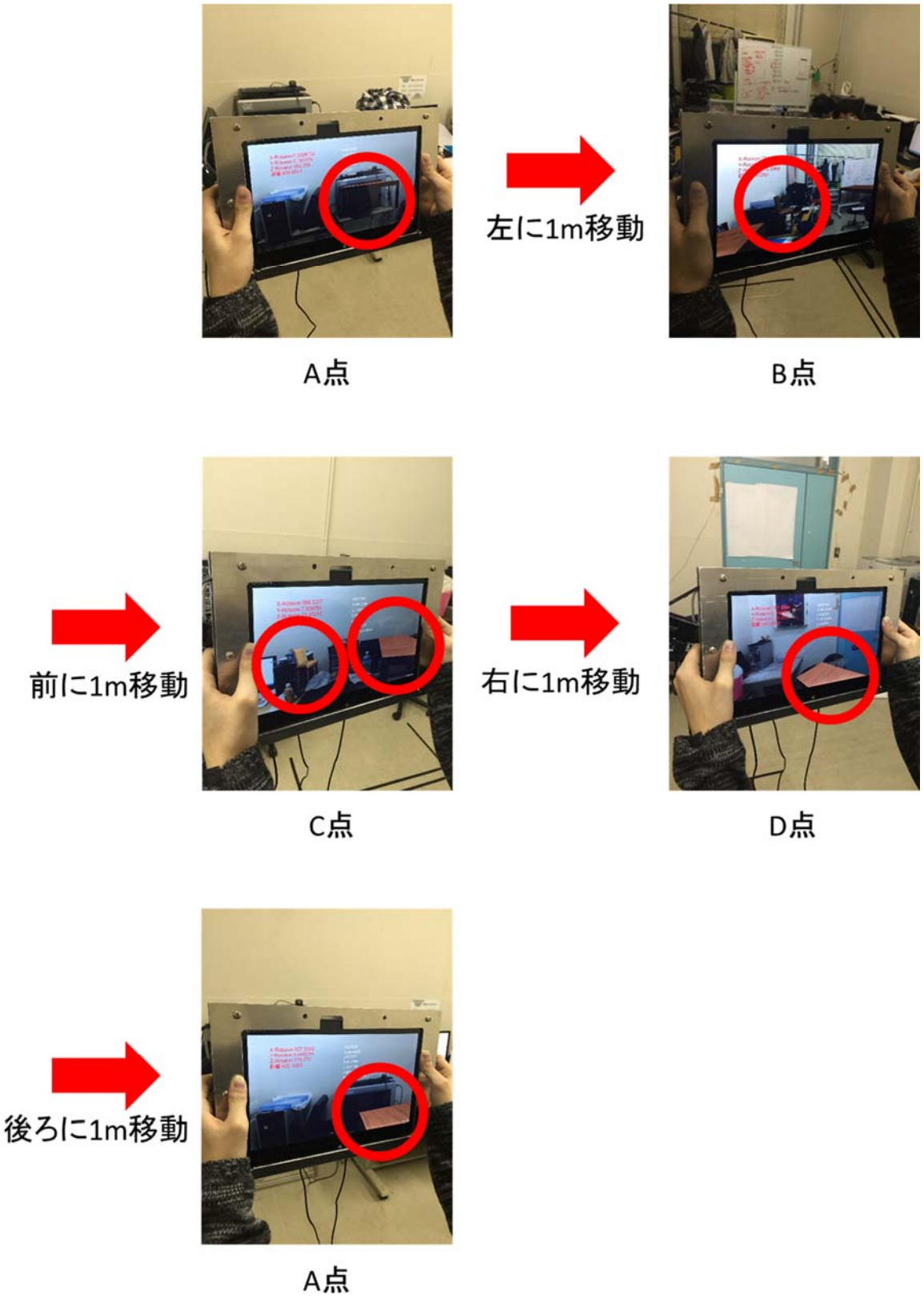

図 5-5　実験結果

5.4 考察

実験の結果から、加速度センサとジャイロセンサが目標となる挙動をせず、加速度センサとジャイロセンサだけでは家具シミュレーションの実現は困難であることが改めて確認できた。

正常に作動しなかった理由を調査したところ、実際の動きとシステム内の動きに違いがあることが判明した(図 5-6)。

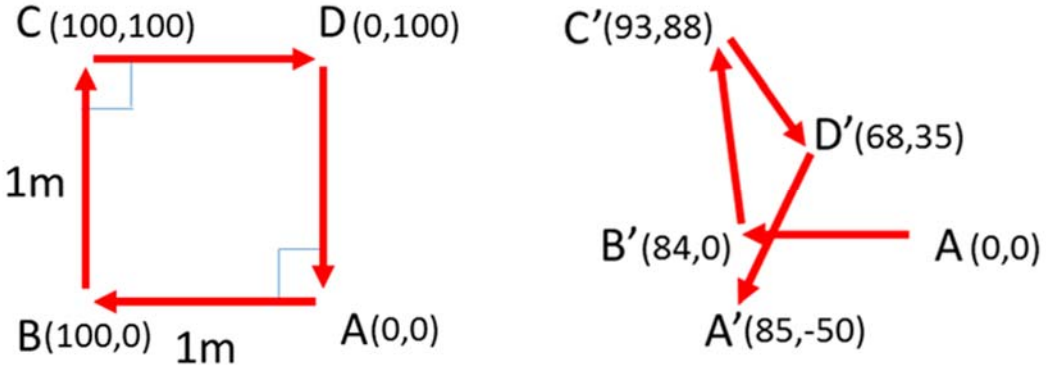

図 5-6 （左）実際の移動
（右）システム内で判定された移動

システム内での移動が、図 5-6 の右図のように判定された理由を、座標点ごとに以下にまとめた。

A 点から B’点・・・比較的正確に動いている。

B’点から C’点・・・y 軸方向の動きは比較的正確に認識しているが、手振れなどの誤差により、x 軸方向の負の方向にずれている。

C’点から D’点・・・加速度センサの積分プログラムの問題により、通常より小さな移動距離となっている。また、この時にジャイロセンサは 90 度と検知すべきところ、誤作動で 160 度変化していると検知している。

D’点から A’点・・・y 軸方向に移動をしているが、手振れなどの誤差により、x 軸の負の方向にずれている。

調査の結果から、実際の動きとシステム内の動きに違いが生じた最大の理由は、C 点から D 点に移動する際に発生するジャイロセンサの誤作動であることが判明した。この誤作動は、第 4 章の時点では発生していなかったため、理由を調査したところ、第 4 章で行っていた実験では、180 度回転させる実験のみを行い、360 度回転させる実験は行っていないことが判明した。そのため、特定の角度で誤差が発生することに気づくことができなかったのではないかと考えられる。しかし今回の実験では、360 度回転させる実験であるため、特定の角度で誤差が発生する範囲に入ってしまい、今回の誤作動が発生してしまったのではないかと考えられる。誤差は、図　5-7 のように 270 度の方向を向いている状態から 0 度の方向を向く際に発生する。

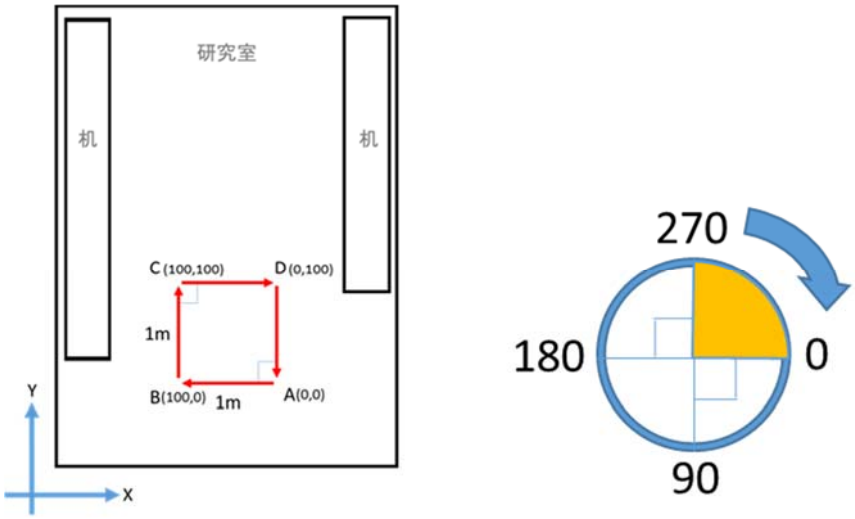

図 5-7　誤差が発生する範囲

C 点で家具の配置が逆に写っていた理由は、C 点で D 点に向いたときに発生したジャイロセンサの誤作動によって、システム内での向きが D 点ではなく D’点に向いてしまっていることが原因であると考えられる。そのため、C 点の映像には A 点と B 点の家具が写り、A 点には椅子、B 点には机が配置されているため、逆に写っているように見えると考えられる。

D 点で机を認識してしまった理由は、実際の位置は D 点であっても、システム内では D’の位置に移動していることが原因であると考えられる。そのため、実際には D 点から A 点を見ていても、システム内では D’点から A’点を見てしまっているため、A’点方向にある B 点の机を認識してしまったと考えられる。

A’の地点まで動いてしまっている状態で、前述の実験結果の D 地点から A 地点に戻ったときに家具を認識できた理由は、システム内の家具のサイズが大きいため、かろうじて認識できたのではないかと考えられる。

第6章 結論 （須多担当）

6.1 目標

我々が製作する家具シミュレーションシステムは、室内で実際に家具を動かすことなく、AR 技術を使用して家具の CG を自由に動かすことにより、室内の景観の変化を模擬できることである。

6.2 目標への達成度

我々は、目標を達成するための条件をすべて満たそうと研究を行った。表 6-1 において、○が達成、△が一部達成、×が達成できなかったことを表している。

表 6-1　目標を達成するための条件

カメラの製作	○
遮光壁の製作	○
フレームの製作	○
レンズ間距離の決定	○
色抽出による距離計測	△
視差画像法による距離計測	△
Windows ストアアプリ形式に出力	○
内蔵カメラの映像出力	○
加速度センサによる移動距離算出	○
ジャイロセンサによる視点移動	○
内蔵センサのみでの家具シミュレーションシステムの開発	○
ステレオカメラを使用した家具シミュレーションシステムの開発	×

機構班は、製作することが困難だと思われていたM17.5のネジ穴を製作し、レンズを調整できるカメラを製作することができた。そして、基板に光が入らないように遮光壁を製作し、カメラに取り付けた。このカメラを2つ製作し、取り付けるためのフレームを製作した。

画像処理班は、機構班が製作したステレオカメラを使用し、適切なレンズ間の距離を実験で決定した。距離を計測するために必要な物体の重心を色抽出することで見つけることができた。また、視差画像を製作することで、大まかな距離を把握することができた。

ソフトウェア開発班は、タブレットPCを使用して、windowsストアアプリ形式に出力することができた。そのため、内蔵カメラの映像を出力することができ、加速度センサとジャイロセンサを使用して移動距離と向きの算出ができた。ステレオカメラのプログラムを統合することはできなかったが、内蔵センサのみを使用した家具シミュレーションシステムを開発することができた。

6.3 問題点

前節にて、家具シミュレーションシステムの動作確認を行ったところ、以下の問題点があげられた。本節では、これらの問題と解決法について記載する。

問題1. ステレオ法による距離計測プログラムが未完成である。

色抽出法、視差画像法の2つの方法で距離計測を試みたが、どちらも具体的な距離を求めるには至らず、どちらが家具シミュレーションに適しているか、判断もできていない。

問題2. 加速度センサで移動させたとき、正しく移動できていない。

実際の移動距離とシステム内の移動距離に誤差が生じている。

問題3. ジャイロセンサで視点移動させたときに、特定の方向で狂う。

実際の視点移動とシステム内の視点移動に誤差が生じている。

問題4. ソフトウェアの動作が重い。

加速度センサとジャイロセンサ、高解像度の内蔵カメラを同時に用いているため処理に時間がかかる。

問題5. 成果物が重い。

今回はアルミを使用して製作を行ったが、重くなってしまったため、新居の見学に来た女性や子供が使用するのは難しい。

6.4 解決策

距離計測Ⅱの研究を進めることで、壁までの距離を測量が可能になり、システムに統合できると考えられる。今回の研究では、距離計測Ⅰに比べ、距離計測Ⅱの方は費やせた時間が少なく、中途半端な結果となってしまった。そのため、深く理解できていない変数が存在し、研究する必要がある。

問題 2 の解決法

移動する際に手振れを検知するため、誤差が生じると考えられる。そのため、加速度センサのソースコードの条件の値を微調整することで、改善できると考えられる。手振れによる誤差移動の検知を減らすには、タブレット PC が動いていないと認識する加速度の範囲を広くする必要がある。これによって、手振れが大きくても横方向に誤差が生じなくなると考えられる。

問題 3 の解決法

使用しているタブレット PC から新しいタブレット PC に取り替えることで、正しい視点移動できると考えられる。実験で使用したタブレット PC は、内蔵ジャイロセンサが特定の方向を認識する時に誤差が生じていることが分かった。私物のタブレット PC で作成したプログラムを実行すると視点移動ができた。内蔵ジャイロセンサだけを取り替えることができないので、新しいタブレット PC に取り替える必要がある。

問題 4 の解決法

使用しているタブレット PC から性能の良いタブレット PC に取り替えることで改善されると考えられる。実験で使用したタブレット PC は、システムを長時間動作させると、負荷がかかり電源が落ちてしまうことがあった。そのため、統合することができなかったステレオカメラによる距離計測まで行うと、処理を行うのが困難だと考えられる。よって、今のタブレット PC では満足にシミュレーションを行うことが難しいため、タブレット PC を変える必要がある。

問題 5 の解決法

使用する素材を、アルミからアクリルに変更することで軽量化できると考えられる。

6.5 総合評価

我々が製作した家具シミュレーションシステムの動作確認を行ったところ、短時間であれば家具の CG を配置し、室内の景観の変化を模擬できることができた。しかし、長時間使用すると、加速度センサの誤差が大きくなり、家具の CG がタブレット PC の画面外に移動してしまう。ステレオカメラのプログラムを統合することができたら、正確な距離を計測できるため、長時間のシミュレーションもできると考えられる。よって、今回の研究では、短時間という条件であれば目標を達成できたと考えられる。

参考文献、URL

[1-1] CEATEC JAPAN 2009――京セラブース、レンズ駆動なしでピントが合うカメラモジュールやモバイル向けソーラーモジュールなどを展示 | K-MAX BLOG
http://kmax.jugem.jp/?eid=597

[1-2] Rubicon 激安 3D スキャナ！！|3D プリンターログ
http://blog.livedoor.jp/printer3d/archives/31765512.html

[1-3] 建物を丸ごと 3D スキャンしてハイクオリティな 3DCG を作り出すエンジン「Thorskan」 - GIGAZINE
http://gigazine.net/news/20140110-thorskan-3d-scan/

[1-4] 機械物理計測講義資料 (10)| 安藤 繁 (東大工計数) ホームページ
http://www.alab.t.u-tokyo.ac.jp/~ando/mesnotea.htm

[1-5] DAVID 3D Scanner | 高精度の 3D レーザースキャナ
http://www.david-laserscanner.jp/

[1-6] 近接 写真測量ソフトウェア OPT Image Based 3D modeling Software
http://www.opt-techno.com/agisoft/

[1-7] 間取り変更で広くなったリビングダイニング 豊中市 | リフォーム | 大阪 | 兵庫 | 奈良 | ナサホーム
http://nasahome.co.jp/works/w_living/plan_living_toyonaka.html

[1-8] 【スマイティ】不動産住宅情報サイト
http://sumaity.com/chintai/tokyo_prop/prop_199972254/

[1-9] しっかり分かる「センサの活用法」: いまさら聞けない加速度センサ入門 (1/2) - EDN Japan
http://ednjapan.com/edn/articles/1205/16/news110.html

[1-10] しっかり分かる「センサーの活用法」: いまさら聞けないジャイロセンサー入門 (1/3) - EDN Japan
http://ednjapan.com/edn/articles/1406/09/news014.html

[1-11] 「自宅に家具を置いてみる」イケアの AR アプリ Page2|WIRED.jp
http://wired.jp/2013/08/23/a-new-ikea-app-lets-you-place-3d-furniture-in-your-home/2/

[3-1] C#/OpenCvSharp でカラートラッキング|画像処理速報
http://opencv.blog.jp/csharp/color_tracking

[3-2] ステレオカメラ作成の道|SlideShare
http://www.slideshare.net/ytanno/ss-21302350

[4-1] Unity で Web カメラを使用する|Lonely Mobiler
http://loumo.jp/wp/archive/20150414000004/

[4-2] oculus rift の加速度センサから頭の変位を出したかった話|nyarurato の製作記録
http://nyaru.blog.shinobi.jp/oculus_rift/20140105

[4-3] 加速度センサの更なる活用 ～入門から応用まで～(Part2)|CEDEC
http://cedec.cesa.or.jp/2009/ssn_archive/pdf/sep1st/PG57_2.pdf

[4-4]Unity + スマホ 5 分で VR - Qiita
http://qiita.com/ELIXIR/items/084839ae0824985fcfaf

謝辞

最後に、この場をお借りして本研究を進めるにあたり、研究テーマやステレオカメラに適したレンズの提供等のご支援いただいたK社、並びに、2年間ご指導ご鞭撻を頂きました金丸隆志准教授、矢崎敬人准教授、研究において親身になり多くの助言を頂いた修士の先輩方、機構の設計や金属加工において快く対応していただいた19号館の皆様方、協力していただいた皆様に心より感謝致します。これをもって謝辞とかえさせていただきます。