

平成27年度 東京理科大学大学院 修士論文

レーザー光を用いた
ハンドジェスチャーの
拡張現実への応用

東京理科大学大学院 理工学研究科 情報科学専攻
明石研究室 修士課程2年

学籍番号 6313640
渡部 祐太

目 次

1 序論	2
1.1 背景	2
1.2 目的	2
1.3 構成	2
2 類似研究	3
2.1 Winding Number による内部外部判定 [3]	3
3 研究概要	6
3.1 概要	6
3.2 提案手法	6
3.2.1 レーザー光デバイス	6
3.2.2 Web カメラを用いたシミュレータデバイス	8
4 アルゴリズム	10
4.1 カメラ画像からの物体データの取得	10
4.2 二次元物体データから三次元データを生成	12
4.3 格子点の内部外部判定	13
4.3.1 $z(k-1), z(k), z(k+1)$ の三点が、この順番で左回りに三角形を構成している場合	13
4.3.2 $z(k-1), z(k), z(k+1)$ の三点が、この順番で右回りに三角形を構成している場合	14
4.3.3 $z(k-1), z(k), z(k+1)$ の三点が、この順番で右回りに三角形を構成していない場合	15
5 実験	18
5.1 環境	18
5.2 Unity について	18
5.3 実行結果	18
6 総括	21
6.1 システムについて	21
6.2 実験結果について	21
6.3 発展	22

1 序論

1.1 背景

我々は、PCを操作する際にマウス、キーボードを使用する。しかしながら、これらのインターフェースを使いこなすのには時間が必要であり、直感的に扱えるインターフェースとは決して言うことはできない。この問題を解決する一例として、ハンドジェスチャーが挙げられる。この技術により、簡単な操作であれば直感的に行うことができるようになり、より多くの人々が扱うことができる。その一方で、ジェスチャーの種類には限りがあるため、操作が複雑になればなるほど、それを純粋なジェスチャーのみで表現することは難しくなってしまう。そこで、近年ではVR技術と組み合わせることにより、この欠点を補う試みが多々見られる。これにより、多くのシチュエーションに対して、ジェスチャーによる操作を応用することが可能になるため、今後、ハンドジェスチャーはより生活に密接した存在になることが予想される。だが、この進歩の途上には一つの新しい問題が存在する。それは、カメラ等の既存のセンサーが使用できない場所での、ジェスチャーの使用が求められる、というものである。具体例を挙げると、トイレのような倫理的にカメラの使用が憚られる場所がある。この場所では、非接触型インターフェースとして、ハンドジェスチャーの需要が高まると予想されるにも関わらず、カメラをセンサーとして使用した既存の技術は使用することができない。そのような状況への対策として、カメラ以外のセンサーを用いたハンドジェスチャー技術が求められている。

1.2 目的

主に二つのことを主眼を置いている。一つ目は、ハンドジェスチャーとVR技術を組み合わせることによって、直感的な操作を行うことができるインターフェースの提案をする、ということである。二つ目は、ジェスチャーを認識するセンサーとして、カメラを使用しないことにより、これまで利用できなかったシチュエーションでもジェスチャーを利用可能にする、ということである。

1.3 構成

本論文は、全六章から構成される。第二章では、本論文において必要となる技術や用語等を紹介する。第三章では、研究の概要と提案手法を記述する。第四章では、提案したシステムの具体的なアルゴリズムを説明する。第五章では、アルゴリズムを実際にどのように実装したのかを、プログラムを交えつつ説明する。第六章では、システムを動作させた結果を提示する。最後に、第七章では、これらのことを行って、考察、結論等の総括を行う。

2 類似研究

VR とハンドジェスチャーを組み合わせる上で、作成した VR オブジェクトとセンサーで認識した物体情報の接触判定が重要となる。本研究では、二次元平面上の図形に対する内部外部判定を利用して、接触か非接触を判定する。そこで、二次元平面の図形の内部外部判定に関する研究を紹介する。

2.1 Winding Number による内部外部判定 [3]

Winding Number というものを用いた内部外部判定方式が存在する。一般的に Winding Number Algorithm と呼ばれるものであるが、これは、内部外部判定したい点 P を中心に、多角形の辺を順番になぞる。この時、点 P の周りを何回回転するかを計算し、その値 (wn) によって内部か外部か判定する。 $wn \geq 1$ の際、多角形は点 P を取り囲んでいると判断できるため、点 P は多角形の内部であると判定される。 $wn = 0$ の際、P は多角形の外側であると判定される。

より具体的な方法を説明する。頂点 $V_0, V_1, \dots, V_n (V_1 = V_n)$ からなる多角形 T において、点 P と頂点 V_i からなる線分 L_i 、点 P と頂点 V_{i+1} からなる線分 L_{i+1} がなす角度の合計を考える（この角度は符号付きで、反時計回りを正、時計回りを負とする）。この時、 θ_i を PV_i, PV_{i+1} のなす角度とすると、一回転は 2π より

$$wn = \frac{1}{2\pi} \sum_{i=0}^{n-1} \theta_i$$

となる。図 2-1 は、点が内部にある場合の例を示している。

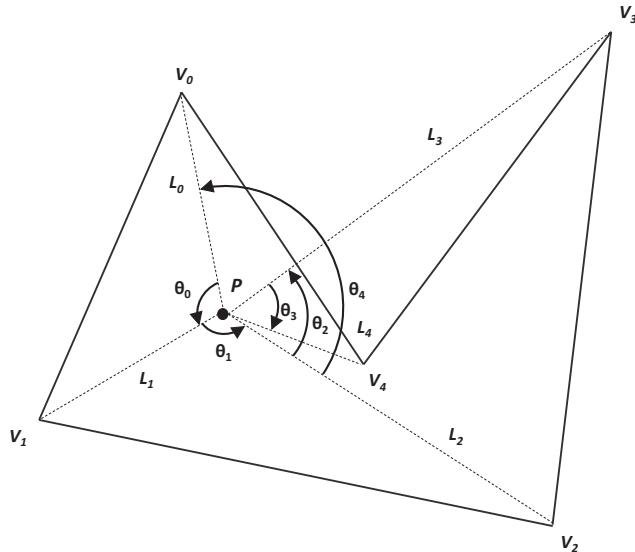


図 2-1. 点が内部に存在する場合

この時、 $\theta_0, \dots, \theta_4$ の合計を考えると、 2π になり、 $wn = 1$ となることが分かる。従って、点 P は内部に存在すると判定される。図 2-2 は、点が外部にある場合の例を示している。

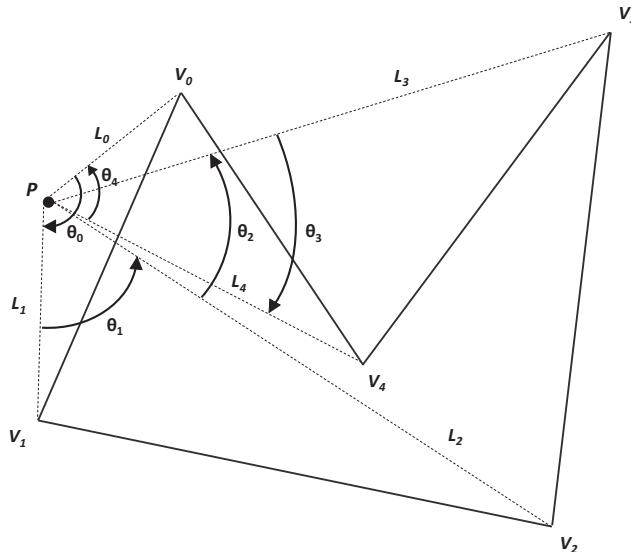


図 2-2. 点が外部に存在する場合

この時、 $\theta_0, \dots, \theta_4$ の合計を考えると、0 になり、 $wn = 0$ となることが分かる。従って、点 P は外部に存在すると判定される。この Winding Number Algorithm を使って、実際に内部外部判定を行ってみる。二次元のスクリーンを用意して、そこに図形の角を描画する。スクリーン上で、正方格子点を考えて、それぞれの格子点が図形の内部か外部かを判定する。結果は、図 2-3 のようになる。



図 2-3. Winding Number Algorithm を用いた内部外部判定

一般に、 $\theta = \cos^{-1}(\cos\theta)$, $\cos\theta = \frac{a \cdot b}{|a| \cdot |b|}$ より、

$$wn = \frac{1}{2\pi} \sum_{i=0}^{n-1} \cos^{-1} \left[\frac{(V_i - P) \cdot (V_{i+1} - P)}{|V_i - P| |V_{i+1} - P|} \right]$$

が成り立つ。この式より、内部外部判定を行えるが、三角関数 $\cos^{-1}\theta$ を使用しているため、計算コストがかかる。

3 研究概要

3.1 概要

ハンドジェスチャーを使用する際に用いられるセンサーとして、一般的とされている Web カメラではなく、レーザー光を用いる。このことは、従来ジェスチャーを使用できなかったシチュエーションや場所においても、使用することを可能とする。

具体的なレーザー光の使用方法を説明する。多数のレーザーと、それと同数の受光器を用意して、お互いが対面するように配置することにより、その間を手が遮った際に検出できるようにする。この検出した情報から手のおおよその形状を認識する。予め 3D オブジェクトを配置した、三次元の仮想空間を用意して、そこに取得した手の形状を描画し、3D オブジェクトを動かすことができるようになる。このことで、レーザー光を用いたデバイスがインターフェースとして使用できることを証明する。

本研究では、これを 2 台の Web カメラを用いて作ったシミュレータデバイスを用意して実験を行う。

3.2 提案手法

3.2.1 レーザー光デバイス

レーザー光を用いたハンドジェスチャーを可能とするため、図 3-1 のようなデバイスを作成した。

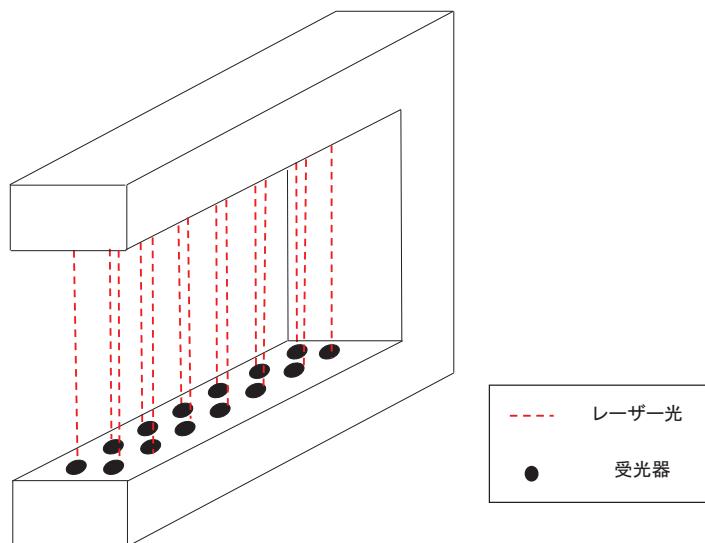


図 3-1. レーザー光を用いたデバイスのイメージ

デバイスの上部に、計 24 個のレーザーを設置する。それと対になるように、下部に同数の受光器を設置する。この間を手が通過すると、レーザー光が遮られたことを受光器が検出する。実際に作成したデバイスを用いて、取得した手情報を画像に描画したものを、図 3-2 に示す。図 3-3 は使用したデバイスの写真である。



図 3-2. レーザー光デバイスを用いて実際に得た情報

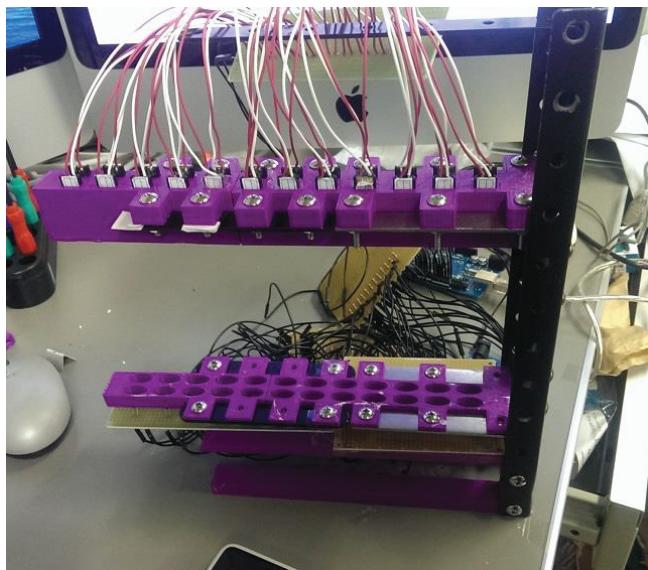


図 3-3. 実際に作成したレーザー光デバイス

図3-2より、レーザー光を用いて手の形状を認識することが可能であることが伺える。しかしながら、このデバイスには問題が存在する。それは、レーザー数が足りないということである。手の情報を得るために、スキャナーのように手を固定した形で通過させる必要がある。このことは動的な手情報を得ることができないことを示している。つまり、VRと組み合わせた直感的なハンドジェスチャーインターフェースを作ることは、不可能である。だが、図3-2の結果から、手の範囲より広くレーザーを設置することが出来れば、動的に手情報を得ることができると推測される。このことを踏まえて、レーザー光の代わりにWebカメラを用いたシミュレータデバイスを作成した。

3.2.2 Web カメラを用いたシミュレータデバイス

図3-4は、レーザー光の代わりにWebカメラを用いたシミュレータデバイスのイメージとなる。側面が一つ空いているボックスの上面と側面に対して、図3-4のようにWebカメラを設置する。レーザー光による検出を再現するために、Webカメラから得られた映像に対して、一定間隔毎に情報を取得する座標を設ける。カメラの対面には、青いスクリーンを貼り付けることにより、間を青以外の物体が遮った時、それを検出できるようにする。実際にボックスの中に手を入れて、取得した情報を図3-5に示す。図3-5は図3-4の上部に設置したWebカメラから取得した情報である。同様の手法で、横のWebカメラからも同時に情報を取得する。縦、横の二つの画像から手情報を三次元空間に構築して、同空間に生成したオブジェクトを動かすことにより、直感的なジェスチャーを可能にするための道筋を示す。図3-6は実際に作成したシミュレータデバイスである。

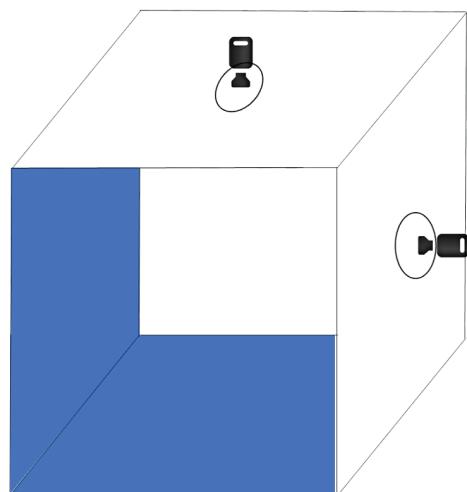


図3-4. Web カメラを用いたシミュレータデバイスのイメージ

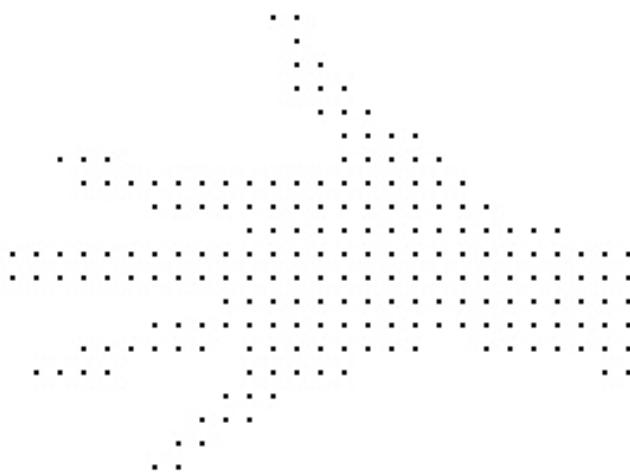


図3-5. シミュレータデバイスから得られた情報

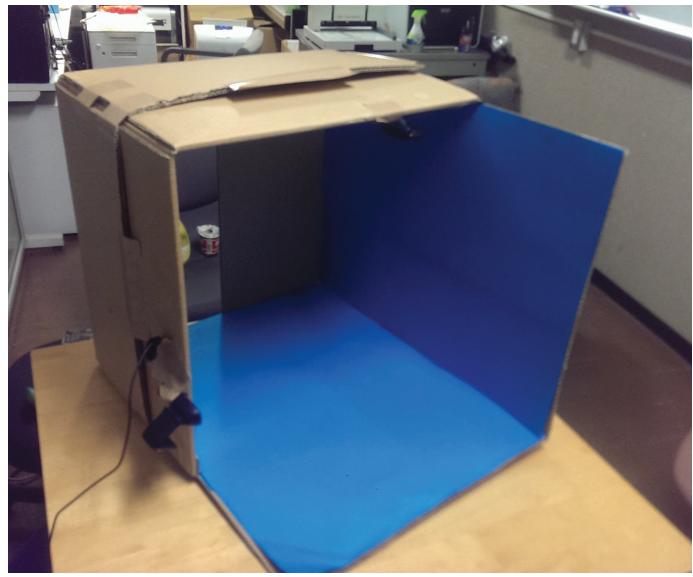


図 3-5. 実際に作成したシミュレータデバイス

4 アルゴリズム

どのような流れで、プログラムが動作しているのか説明を行う。大まかな流れに分けると次のようになる。

1. デバイスに設置した二台のWeb カメラから、それぞれ映像をキャプチャする。これらのカメラは縦、横に設置される。
2. それぞれのキャプチャした画像上に、正方格子点が存在すると考える。点と重なる画素について、青色かどうか判定して結果を取得する。青色でない場合、何かしらの物体が存在すると判断する。
3. 縦、横、それぞれの結果を組み合わせることで、3次元のデータを生成する。
4. 三次元空間に、正方格子点が存在すると考える。この空間には、あらかじめデータを入力した三次元オブジェクトを配置している。すべての格子点について、このオブジェクトの内部にあるか外部にあるのかを判定して結果を取得する。
5. 3で得た物体情報が格納された三次元データと、5で得た内部外部判定が格納された三次元データを比較する。3で物体が存在すると判断された点と、4で内部と判定された点が重なっている場合、物体はオブジェクトの内部に存在すると判断する。
6. 物体がオブジェクト外部に出るよう三次元オブジェクトを移動させる。

本章では、2、3、4についてより詳細な説明を行う。

4.1 カメラ画像からの物体データの取得

縦に設置した Web カメラから得た画像について述べる。まず、Web カメラから取得した画像は RGB 色空間での表現となっているため、特定の色が抽出しやすい HSV 色空間への変換を行う。



図 4-1. RGB 空間での表現

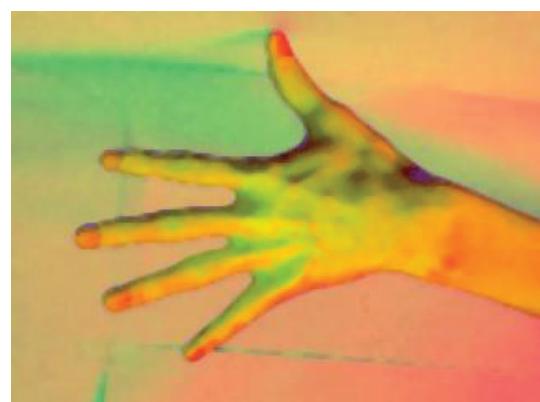


図 4-2. HSV 空間での表現

図 4-1 は RGB 色空間で表現した画像、図 4-2 は HSV 色空間で表現した画像となっている。変換後の画像上で、正方格子点を考える。格子点を画像上に可視化したもののが図 4-3 となる。

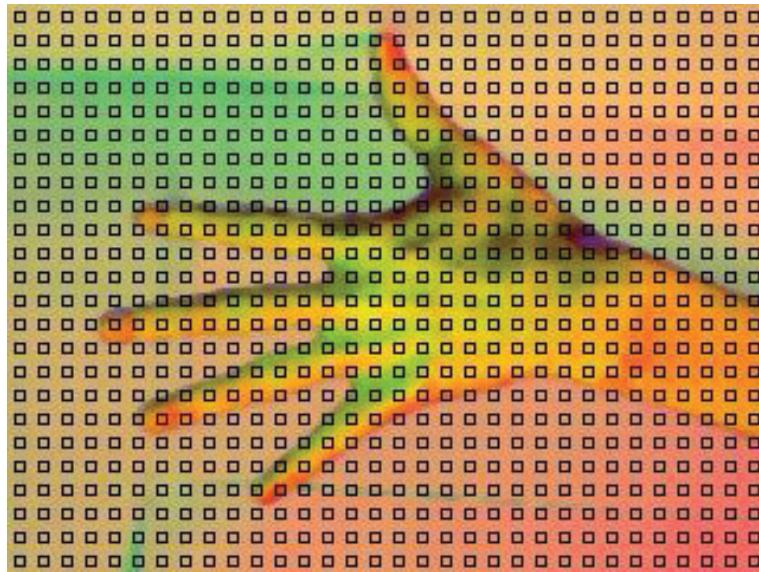


図 4-3. 格子点のイメージ

これらの格子点と重なっている画素について、青色かどうかを判定する。図の 4-4 は、青色ではない部分を描画したものである。この一連の流れで、レーザー光を用いた際と同じデータを取得することができる。これを横についても、同じことを行う。

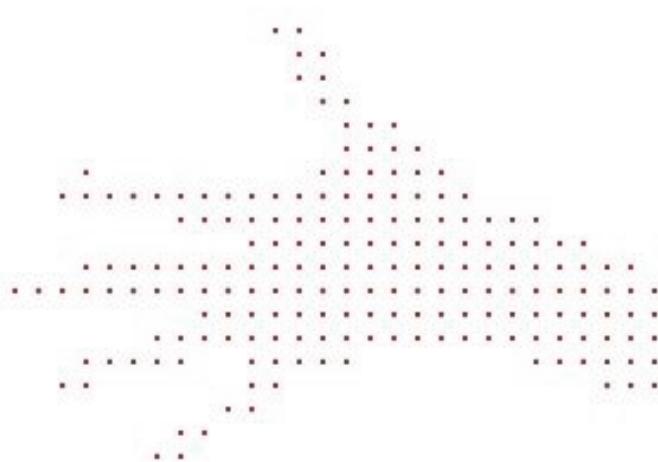


図 4-4. 青色ではない部分を描画

4.2 二次元物体データから三次元データを生成

Web カメラから取得したデータから三次元データを生成する。図 4-5、図 4-6 はそれぞれ縦に設置したカメラ、横に設置したカメラから得たデータとなっている。



図 4-5. 縦に設置したカメラから得た
データ

図 4-6. 横に設置したカメラから得た
データ

始めに、横のカメラから得たデータを元に、 $Y=0$ から物体が存在するという情報があるか検索する。存在する情報が発見された時、 X の値を一旦保存する。縦のカメラから得たデータにおいて、先ほど保存した X の値を用いて、物体が存在する情報がないのか検索する。存在する情報が発見された時、 Z の値を保存する。この時の、 X 、 Y 、 Z には物体が存在すると判定する。これをすべての X 、 Y 、 Z について繰り返し行うことで、三次元のデータを生成する。実際に、図 4-5、図 4-6 から生成した三次元のデータが、図 4-7 となる。

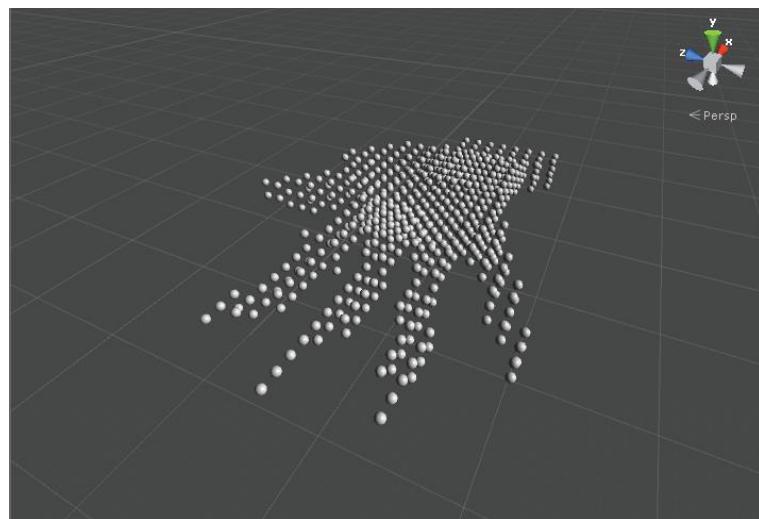


図 4-7. 生成した三次元データ

4.3 格子点の内部外部判定

オブジェクトが存在する三次元空間に、正方格子点が存在すると考える。それらの点がオブジェクトの内部にあるか、外部にあるのかを判定する。三次元空間を一定間隔の層に分けて、各階層毎に内部外部判定を行うことで、擬似的に三次元オブジェクトに対して内部外部判定を行うことができる。各階層毎の内部外部判定には、外積計算と Cauchy の積分定理を組み合わせた手法を用いる。

ここで、外積計算による判定方法と、Cauchy の積分定理による判定方法について説明を行う。一般的な特徴として、外積計算による方法は、くぼみや穴などを有するドーナツ型の図形に対する内部外部判定が不利であり、かつ複雑形状の図形において、境界線から離れた内部で判定が困難である。Cauchy の積分定理による方法は、くぼみや穴などを有するドーナツ型の図形に対する内部外部判定が有利である一方で、複雑形状の図形において、境界線付近での判定に難があるというデメリットが存在する。そこで Cauchy の積分定理の弱点を、外積計算による方法で補うことを考える。しかしながら、外積計算による方法は、複雑形状における内部外部判定で誤判定が引き起こすことが多い。そこで外積計算計算による方法の改良版を使用する。既存の方法では、調査点 P と多角形の輪郭を構成する 2 点 $z(k), z(k+1)$ の合計三点を用いて計算を行っている。改良版は、調査点 P と輪郭を構成する三点 $z(k-1), z(k), z(k+1)$ の合計四点を用いて計算を行う。

4.3.1 $z(k-1), z(k), z(k+1)$ の三点が、この順番で左回りに三角形を構成している場合

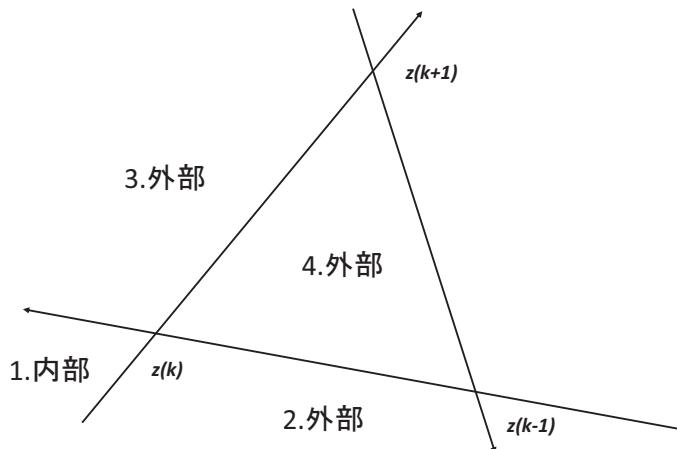


図 4-8. 三点が左回りの三角形

1. ベクトル $pz(k-1)$ とベクトル $z(k-1)z(k)$ の外積の値が 0 以上、かつベクトル $pz(k)$ とベクトル $z(k)z(k+1)$ の外積の値が 0 以上ならば内部と判定する。
2. ベクトル $pz(k-1)$ とベクトル $z(k-1)z(k)$ の外積の値が 0 以上、かつベクトル $pz(k)$ とベクトル $z(k)z(k+1)$ の外積の値が負ならば外部と判定する。

3. ベクトル $pz(k-1)$ とベクトル $z(k-1)z(k)$ の外積の値が負であり、かつベクトル $pz(k)$ とベクトル $z(k)z(k+1)$ の外積の値が0以上ならば外部と判定する。
4. ベクトル $pz(k-1)$ とベクトル $z(k-1)z(k)$ の外積の値が負であり、かつベクトル $pz(k)$ とベクトル $z(k)z(k+1)$ の外積の値が負ならば外部と判定する。

4.3.2 $z(k-1), z(k), z(k+1)$ の三点が、この順番で右回りに三角形を構成している場合

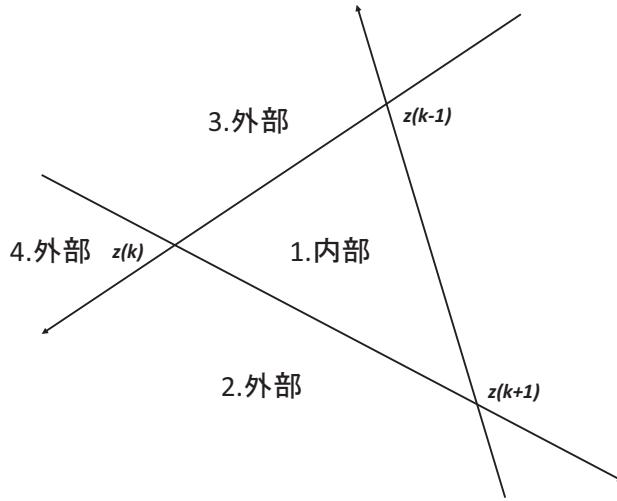


図 4-9. 三点が右回りの三角形

1. ベクトル $pz(k-1)$ とベクトル $z(k-1)z(k)$ の外積の値が0以上、かつベクトル $pz(k)$ とベクトル $z(k)z(k+1)$ の外積の値が0以上ならば内部と判定する。
2. ベクトル $pz(k-1)$ とベクトル $z(k-1)z(k)$ の外積の値が0以上、かつベクトル $pz(k)$ とベクトル $z(k)z(k+1)$ の外積の値が負ならば外部と判定する。
3. ベクトル $pz(k-1)$ とベクトル $z(k-1)z(k)$ の外積の値が負であり、かつベクトル $pz(k)$ とベクトル $z(k)z(k+1)$ の外積の値が0以上ならば外部と判定する。
4. ベクトル $pz(k-1)$ とベクトル $z(k-1)z(k)$ の外積の値が負であり、かつベクトル $pz(k)$ とベクトル $z(k)z(k+1)$ の外積の値が負ならば外部と判定する。

4.3.3 $z(k-1), z(k), z(k+1)$ の三点が、この順番で右回りに三角形を構成していない場合

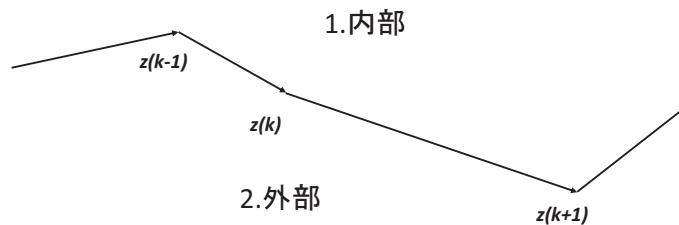


図 4-11. 三点が三角形を構成していない

1. ベクトル $pz(k)$ とベクトル $z(k)z(k+1)$ の外積の値が 0 以上ならば内部と判定する。
2. ベクトル $pz(k)$ とベクトル $z(k)z(k+1)$ の外積の値が負ならば外部と判定する。

この場合、ベクトル $pz(k)$ とベクトル $z(k)z(k+1)$ の外積の値の代わりに、ベクトル $pz(k-1)$ とベクトル $z(k-1)z(k)$ の外積の値を用いることも可能である
実際にこのアルゴリズムを使用して、内部外部判定を行った結果が図 4-11 となる。

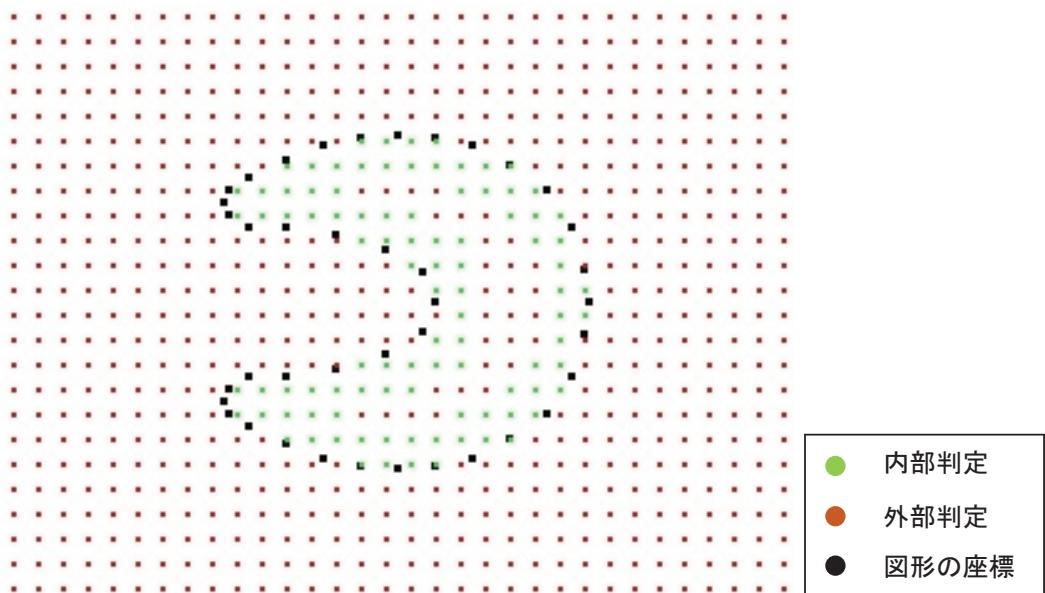


図 4-11. 改良版の外積計算による内部外部判定

図を見ると、境界線付近にて正しく判定できていることが分かる。しかしながら、境界線から離れた内部の格子点が正しく判定されていない。

Cauchy の積分定理による方法は、くぼみや穴などを有するドーナツ型の図形に対する内部外部判定が有利である一方で、複雑形状の図形において、境界線付近での判定に難があるというデメリットが存在する。実際に Cauchy の積分定理によって内部外部判定を行った結果が、図 5-9 となる。

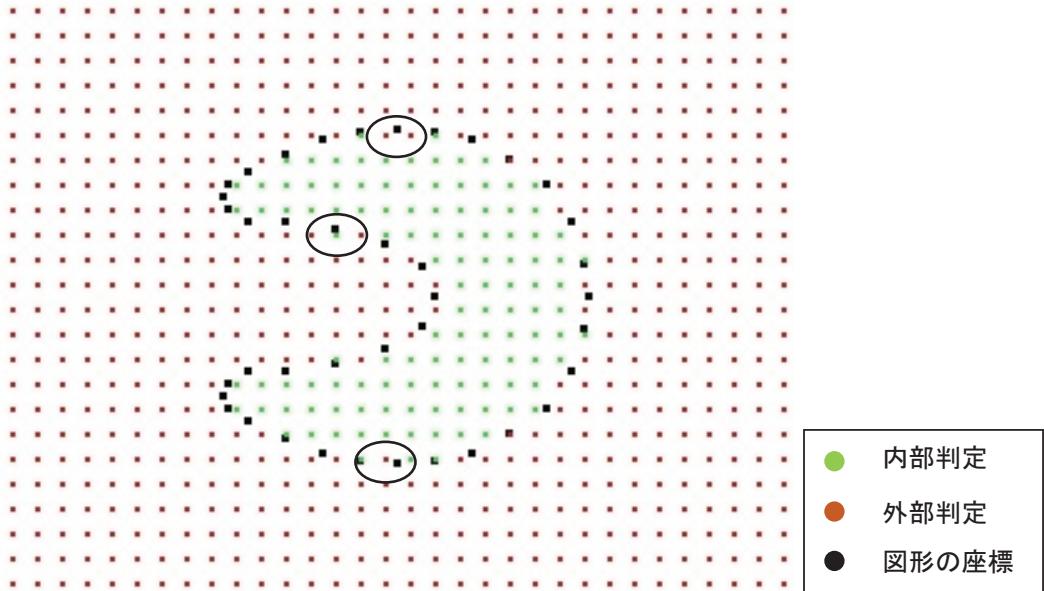


図 4-12. Cauchy の積分定理による内部外部判定

図を見ると、円で囲まれた格子点が正しく判定されていないことが分かる。

以上の結果を踏まえて、それぞれの方法のデメリットを補うように組み合わせた手法を用いることにする。具体的には、境界線付近の領域では、改良版の外積計算による方法を使用し、境界線から離れた領域では Cauchy の積分定理による方法を使用する。この手法をもとに、内部外部判定を行った結果が、図 4-10 となる。



図 4-13 外積計算と Cauchy の積分定理を組み合わせた判定

改良版の外積計算、および Cauchy の積分定理のそれぞれのデメリットが上手く補われて、すべての格子点について正しく判定されている。同様の処理をすべての階層で行うことで、三次元空間の格子点に対して内部外部判定を行う。

5 実験

5.1 環境

OS	Windows 8.1 Enterprise 64bit
CPU	Intel(R) Core(TM) i7-2600 CPU 3.40GHz
メモリ	4.0GB
統合開発環境	Unity 5.1.0, Visual Studio Community 2015
開発言語	C#
ライブラリ	OpenCV 2.4.10
ウェブカメラ	Logicool HD Webcam C270, Logicool HD Webcam C525

5.2 Unity について

Unity とは、Unity Technologies が開発を行っている、統合開発環境を内蔵しているゲームエンジンである。ウェブプラグイン、デスクトッププラットフォーム、ゲーム機、携帯電話用 OS など、多岐にわたるプラットフォームに対応している。スクリプト言語として、C#、Javascript、Boo を使用することができる。3D 物理演算を容易に扱うことができるのが特徴である。

5.3 実行結果

Unity を用いて、3D 空間にオブジェクトを配置する。実際にオブジェクトを配置した結果が図 5-1、図 5-2 となる。

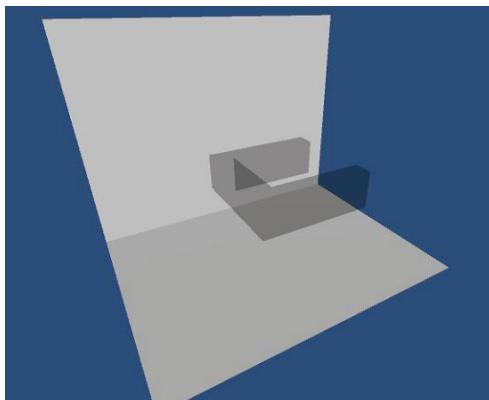


図 5-1. 作成した 3D オブジェクト

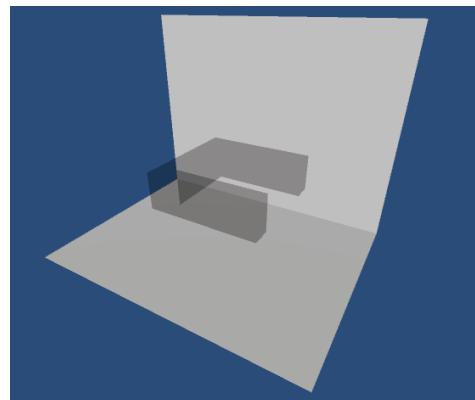


図 5-2. 作成した 3D オブジェクト

このオブジェクトに対して、3D 格子点の内部外部判定を行う。結果は、図 5-3 となる。

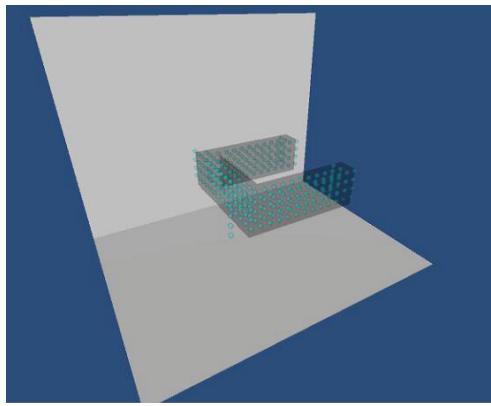


図 5-3. 内部外部判定結果

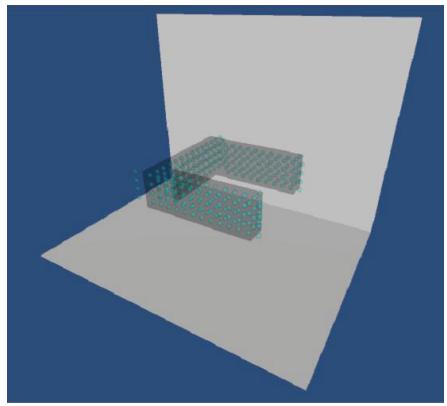


図 5-4. 内部外部判定結果

青緑色の球が、内部と判定された格子点となる。若干の誤判定はあるが、概ね見た目通りに内部外部を判定できていることがわかる。ここで、3D オブジェクトの窪んだ部分に手を挿入することで、内部外部判定を用いた当たり判定が正常に動作していることを確認する。

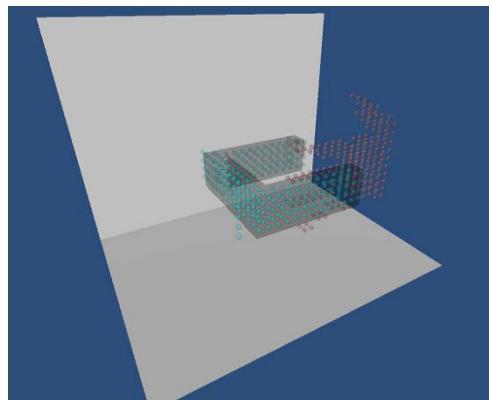


図 5-5. 手をオブジェクトの窪みに挿入

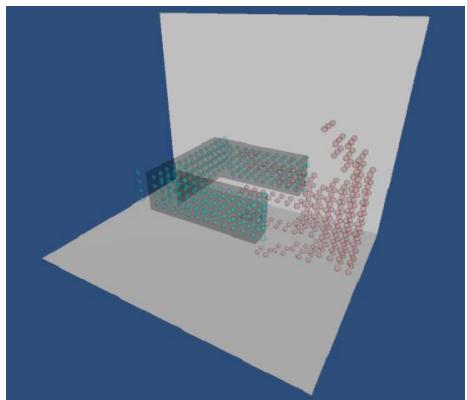


図 5-6. 手をオブジェクトの窪みに挿入

図 5-5、図 5-6 の赤い点で描写されているものが、手と認識された情報を可視化したものである。窪んだ部分に手を挿入しても、3D オブジェクトが移動しないことが確認できた。この状態で、手前に動かした結果を図 5-7、図 5-8 に示す。

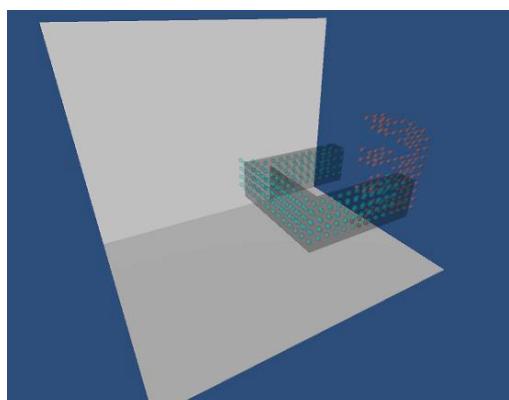


図 5-5. 手を手前に移動

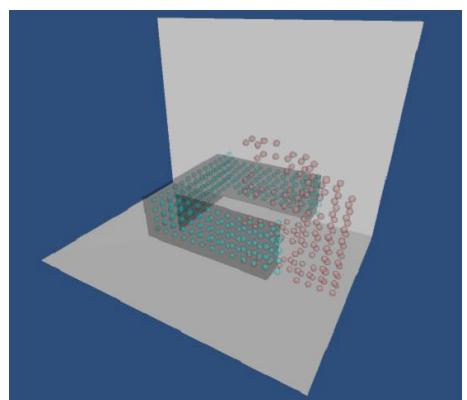


図 5-6. 手を手前に移動

手がオブジェクトに触れるのと連動して、3D オブジェクトも手前に移動するこ

とが確認できた。これらの結果から、内部外部判定を用いた接触判定は正常に動作していることが確認できる。

6 総括

6.1 システムについて

本研究での目的を改めて述べる。一つは、ハンドジェスチャーとVR技術を組み合わせることによって、直感的な操作を行うことができるインターフェースの作成である。二つ目は、カメラを使用しないことにより、これまで利用できなかったシミュレーションでもジェスチャーを利用可能にする、ということである。

二つ目の、カメラを使用しないジェスチャーの利用についてだが、十分に要件を満たしていると考えている。今回使用したシミュレータデバイスだが、実際の光レーザーを使用したデバイスのデータと、ほぼ同じ形でデータを取得することができる。すなわち、4章の図4-3の格子点と同じ数だけのレーザー、および受光器を用意することができれば、Webカメラによるシミュレータでなくとも、今回の実験と同様の結果が得られることが推察できる。しかしながら、必要となるレーザー光を計算すると、縦、横それぞれにおいて、 32×24 個を使用することになり、合計で1536個のレーザー光を使用することになる。これを一個人で用意することは難しいため、実際にデバイスを作成しようとすると、一企業単位の組織が必要となる。

一つ目の直観的な操作ができるインターフェースについてだが、今回実現できたとは言うことはできない。その原因だが、内部外部判定を行う際の、計算速度が問題としてあげられる。本システムでは、オブジェクトと手が重なった際に、オブジェクトを移動させて、再度内部外部判定を行っている。この内部外部判定の計算に時間がかかるため、あまりにも早く手を動かすと、計算が追い付かなくなり手がオブジェクトに食い込んでしまう。これにより、オブジェクトの挙動が想定外のものとなってしまう。これを解決するためには、より早い内部外部判定アルゴリズムを導入する必要があると考えられる。

6.2 実験結果について

概ね想定通りの結果を示したが、図6-3、6-4での内部外部の誤判定は想定外のものとなった。同様のアルゴリズムを用いた、第5章の図5-13では、実験で使用した図形より複雑なものにも関わらず、内部外部の誤判定は起こらなかったからである。この誤判定は図形の境界線付近で起こっているため、改良型の外積計算によって生じている。おそらく、入力される図形が第5章の図5-13のようにある程度複雑な図形の場合は、このような誤判定は起こらないが、今回の実験のように、あまりにもシンプルすぎる図形の場合、誤判定が起こると予想される。本研究の最終目標はVR技術との組み合わせで、インターフェースを作ることである。従って、三点のみで構成されている、あまりにもシンプルすぎる図形を想定していない。ある程度複雑な図形を想定しているため、全て第4章、4.3.3の計算を適用している。そのことが、今回の誤判定につながったと考えられる。

6.3 発展

三次元オブジェクトの内部外部判定を行う際に、層毎に分けることで二次元の内部外部判定を行っている。この手法でも、十分な判定結果を得られるが、三次元を三次元のまま考えて内部外部判定を行う手法を用いると、同様かそれ以上の結果を得られると考えられる。従って、より正確な結果を得るためにも、今後はそのような手法について模索するべきだと考えられる。しかしながら、計算量が大幅に増えることは確実であり、これを解決するための手段も同時に考えなければならない。

参考文献

- [1] 非凸形状を有する平面図形の形状認識に関するベクトル解析的手法の問題点.,
数理解析研究所講究録 第 1544 巻 2007 年 8-12
- [2] 児玉賢史, 複雑形状認識の問題点と対処方法, 数理解析研究所講究録, 第 1643
巻 2009 年 148-156
- [3] Inclusion of a Point in a Polygon - Geometry Algorithms,
http://geomalgorithms.com/a03_inclusion.html