平成１４年度　卒業論文

マーチングキューブズ法による
３Ｄ構築における法線ベクトルの平均化

奈良女子大学理学部情報科学科

数理情報学講座　和田研究室

藤原　左知子

１．概要

本研究では、DeltaViewer [3]における３Ｄ 構築アルゴリズムをマーチングキューブズ法で行い、そして法線ベクトルの平均を行って面と面の境界を滑らかに見えるようにした。DeltaViewer は２次元断面画像から任意の境界を抽出して曲面データを３次元構築し、それを OpenGL を用いてコンピューター画面上に表示するソフトウェアである。DeltaViewer における従来の３Ｄ構築アルゴリズムでは描画しときに面と面の境界がくっきりと見えていた。それは実行結果である図１- ａ を見ればわかる。そして本研究の実行結果である図１- ｂ から面と面の境界を滑らかに見えるようにするという目的が達成できたことがわかる。

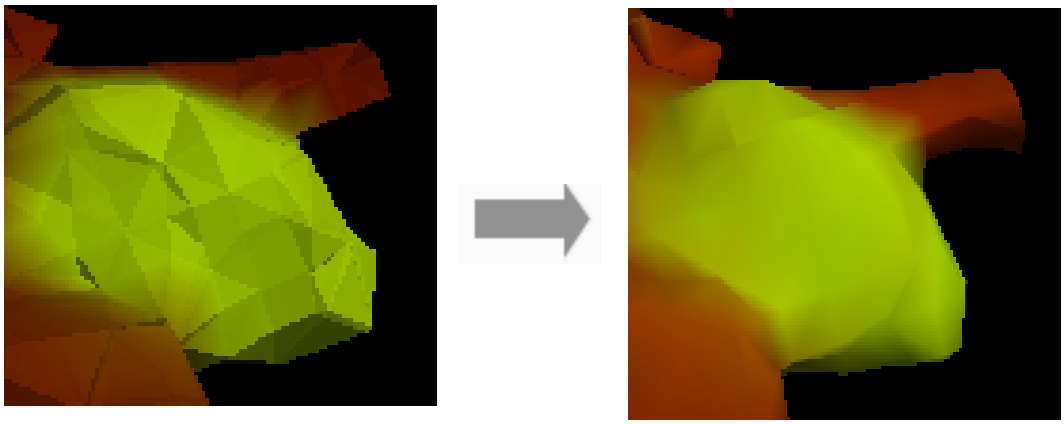

図１- a:DeltaViewer の従来の実行結果
図１- b:本研究の実行結果

描画したときにできる面と面の境界を滑らかに見えるようにするために、本研究では隣り合う面同士で法線ベクトルの平均化を行った。OpenGL でポリゴンの描画を行う際、ポリゴンの頂点とその頂点における法線ベクトルは同時に指定することが必要である。そのため本研究で重要なことは、ポリゴンを描画する前に全ての頂点における法線ベクトルを隣り会う面同士で平均化しなければならないということだ。しかし、DeltaViewer における従来の３Ｄ 構築アルゴリズムでは、１つのポリゴンごとに法線ベクトルを計算して描画する作業を繰り返していたので、このまま法線ベクトルの平均化アルゴリズムだけを加えることは困難であった。そこで、本研究では新たな３Ｄ 構築アルゴリズムを考えた。そのアルゴリズムは、本文で詳しく述べる。

目次

２．研究目的

DeltaViewer における従来の３Ｄ構築アルゴリズムでは、描画されたときにポリゴンとポリゴンの境界が図２のようにくっきりと見えていたので、本研究ではその境界が滑らかに見えるようにしたいと考えた。そこで必要となるのが法線ベクトルの平均化である。従来の３Ｄ 構築アルゴリズムに本研究で必要となる法線ベクトルの平均化アルゴリズムを加えようと考えた。しかし、本研究では頂点の指定までに法線ベクトルを平均する計算が必要になるのでこの方法は困難であった。そこで本研究では新たな３Ｄ構築アルゴリズムが必要となった。

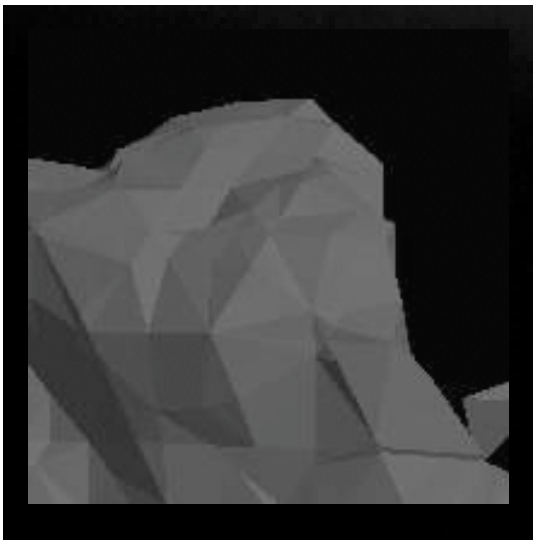

図　２　：　DeltaViewer における従来の３Ｄ構築アルゴリズムでの実行結果の一部

３．OpenGL におけるポリゴンの描画

ここでは OpenGL におけるポリゴンについて記述する。ポリゴンとは多角形、つまり環状の線分によって閉じられた領域のことで、通常ピクセルで内部を埋め尽くして描画するものである。OpenGL においてポリゴンは凸型であることが必要で、非凸型や同一平面上にない頂点でのポリゴンは正確に描画することができない。そこで、３角形なら３つの頂点が常に同一平面上に位置し、常に凸型であるので正確にポリゴンを描画することができるため、DeltaViewer ではすべてのポリゴンを３角形で描画することにしている。

ポリゴンは頂点座標を指定して描画するのはもちろんだが、頂点とともに法線ベクトルも指定している。照光機能を使用するために法線の指定が必要なのである。法線とは、その面に対して垂直なベクトル、つまり面の向きのことである。OpenGL で法線ベクトルの指定ができるのは、ポリゴンの頂点だけで、ポリゴンを描画する時には、頂点の指定と法線ベクトルの指定は同時に行わなければならない。法線ベクトルの算出方法については「５.１　法線ベクトルの算出方法」でくわしく述べる。

ポリゴンには表と裏があり、それは頂点に指定する法線ベクトルの向きを指定することによってどちらを表にするかを決める。表のみ描画したポリゴンはちょうどマジックミラーのようで、表からはポリゴンを見ることができるが裏からは何もないように見えてしまう。もし、ポリゴンの表を描画するはずが間違えて裏を描画するように法線ベクトルを指定してしまうと３Ｄ 構築したときのポリゴンの向きがそこだけ反対となるので、穴の空いたような立体が構築されてしまう。表と裏の両面を描画すれば法線ベクトルの向きは関係なくなるので穴が空くことはないのだが、裏面を描く分だけ描画するのに時間を要するので時間短縮のために表だけを描画したい。そのため法線ベクトルの向き、つまりは頂点の指定する順番を統一することはとても重要なのである。

４．　３D 構築アルゴリズム

本研究では３D 構築アルゴリズムをマーチングキューブズ法で行い、ポリゴンの頂点となる座標をすべて求め、それらの頂点座標からポリゴンごとに法線ベクトルを計算した。そして、１つの頂点における法線ベクトルが複数存在するので平均して１つにした。このような準備を行った後に OpenGL を用いてコンピュータ上に描画するのであるが、まずこの章ではマーチングキューブズ法、これを用いて描画するポリゴンについて詳しく述べる。

４.１　マーチングキューブズ法

マーチングキューブズ法とは、３次元データを小さな立方体に分割してボクセル値が等しい面（等値面）をポリゴンに変換し、変換されたポリゴンをレンダリングする手法である。以下にマーチングキューブズ法における３D 構築の方法を記述する。

１．　３次元データから小さな立方体に分割する。（図３- a）

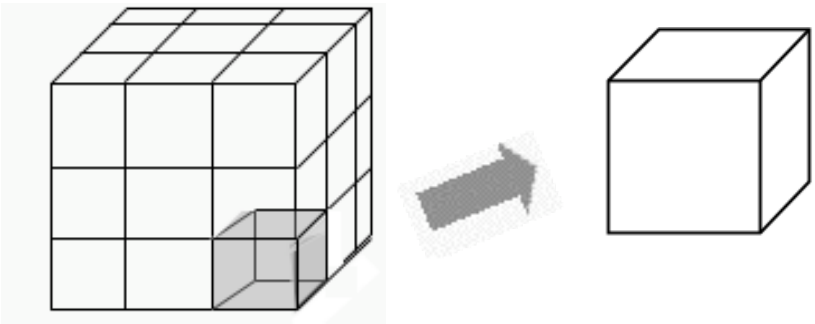

図３- a

２．　１.で分割した立方体の８頂点についてピクセル値を調べていく。ピクセル値とはその頂点における三次元データの色の値である。DeltaViewer ではピクセル値は－１から＋１までのスケールで double 型と、境界となる値を「０」とする。ここでは図３- b のように、頂点のピクセル値が境界値よりも大きければ「＋」、小さければ「－」というように表現しておく。

３．　２.で求めた頂点のピクセル値から立方体の１２個の返上で境界を探す。境界値は「０」なので辺の両端にある頂点のピクセル値が「＋」と「－」の間に境界があるので境界点となる座標を求める。図３- ｂ のような場合なら境界点は図３- ｃ のように３つ存在することになる。このとき、「３．OpenGL におけるポリゴンの描画」からもわかるようにポリゴンの向きは非常に重要なので頂点座標の指定する順番を統一しておく。

４．　３.で求めた境界点を指定してポリゴンを描画する。（図３- d）

５．　２.～４.の作業を１.で分割した立方体すべてにおいて行うと、ポリゴンとポリゴンが繋がり３D 構築することができる。

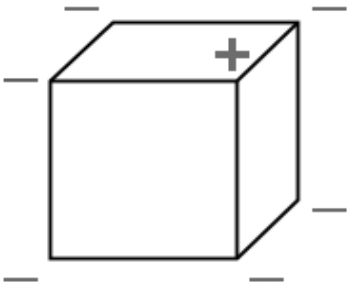

図３- b

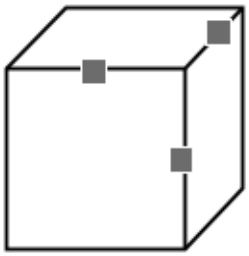

図３- c

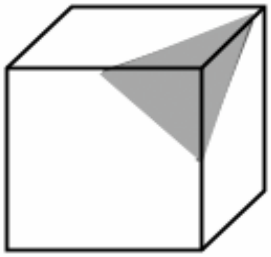

図３- d

４.２　境界の引き方

マーチングキューブズ法（４.１参照）でポリゴンを作るとき、辺上に作った境界点同士で引いた線が描画するポリゴンの辺であり、また境界線であり、これは立方体の面上にできるので、面について境界の引き方を考える。ここでも、面における４頂点のピクセル値が境界値よりも大きければ「＋」、小さければ「－」というように表現すると図３- e のような境界を面に引くことができる。

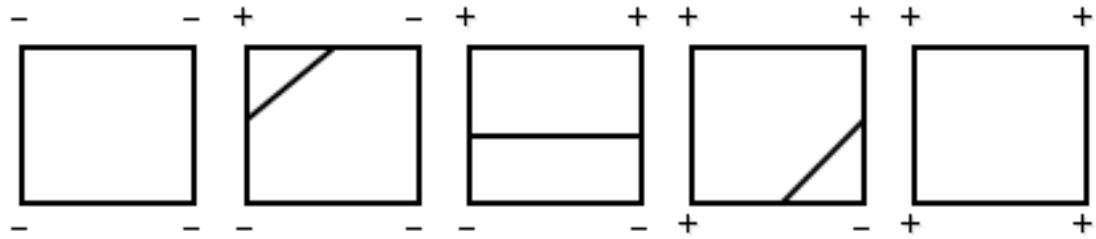

図３-ｅ

図３-ｅの例では１通りしか境界の引き方はないのだが、４頂点のピクセル値のうち向かい合う２つの頂点が「＋」で残りが「－」であった場合、境界を２通り引くことができる。これは統一して場合けしておかないと３Ｄ構築したときにポリゴン同士が繋がらなくなってしまうので注意が必要である。そこで、統一する方法は面の中心座標におけるピクセル値が、境界とするピクセル値よりも大きいときと小さいときで場合けし、図３-ｆのように境界を引くことにした。

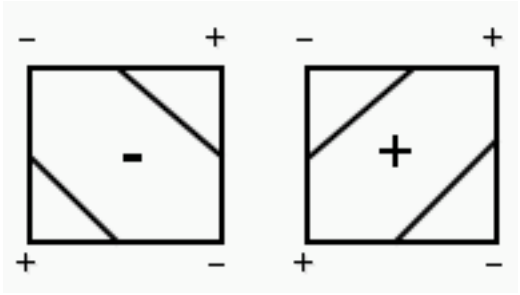

図３-ｆ

このように境界の引き方がわかれば立方体に描くポリゴンがどのような形になるのかがわかるので、それを３角形に分割し、３角形の頂点が右回りとなるように辺上にある境界点の順番を指定する。そのために図３-ｇのように辺に０～１１の番号をつけ、ポリゴンの頂点が立方体のどの辺上にあるのか図３-ｇの辺の番号を右まわりとなるように指定する。

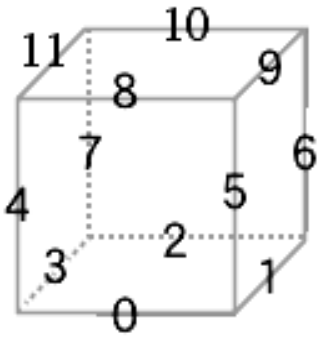

図３-ｇ

４.３　ポリゴンの形の場合分け

１つの立方体に対してどのようなポリゴンを描画するかを判別するために、フラッグを使用する。まず、図３- h のように立方体の頂点に０～７の番号をつけ、それに対応させた配列 mT[0]～mT[7]を用意する。そして頂点０～７におけるピクセル値を mT[0]～mT[7]に格納し、それぞれ境界値より大きいか小さいかによって場合分けをすると２^{8} ＝２５６通り存在し、これを switch 文で場合分けする。まず、セレクタ（ switch(a){}とするときの「a」のこと）を考える。８頂点におけるピクセル値が境界値よりも小さければ「０」、大きければ「１」（00010001 のように）８桁で表したい。しかし、セレクタは整数型でないといけないのでこの８桁の数を２進数として考え、これを１０進数に変換したものをセレクタとすると以下のように２５６通りの場合分けが可能となる。

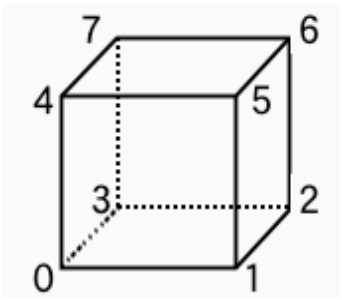

図３- h

（頂点　7　6　5　4　3　2　1　0）
（　　0　0　0　0　0　0　0　0）$_2$＝（0）$_{10}$
（　　0　0　0　0　0　0　0　1）$_2$＝（1）$_{10}$　　（∵2^0＝1）
（　　0　0　0　0　0　0　1　0）$_2$＝（2）$_{10}$　　（∵2^1＝2）
　　：　　：
（　　1　1　1　1　1　1　1　1）$_2$＝（255）$_{10}$
（∵ $2^0+2^1+2^2+2^3+2^4+2^5+2^6+2^7$＝ 255）

そこでピクセル値が境界値よりも大きければ「１」のフラッグをたてるために OR 演算（｜）を使用した。OR 演算は表１のようにどちらかのビットが１であれば１となる。よって最初は（０００００００００）$_2$として、ピクセル値が境界値より大きいビットで、０|１　＝１の計算を行い、フラッグをたてることにした。

実際に OR 演算子を用いて int 型である「 mFlag」にフラッグをたてて、２５６通りのセレクタとするプログラムの例、実行結果を以下に示す。

OR演算

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

表１

＜プログラム例＞

```
// 頂点のピクセル値を格納
    mT[7]= 0.5;
    mT[6]= －0.5;
   mT[5]= 0.5;
    mT[4]= －0.5;
    mT[3]= 0.5;
    mT[2]= －0.5;
    mT[1]= 0.5;
    mT[0]= －0.5;

//  フラッグをたてる
//  境界値のピクセル値は「０」
    mFlag=0;        //  ２進数で（0000 0000）
    if(mT[7]>0)   mFlag |= 128;
    if(mT[6]>0)   mFlag |= 64;
    if(mT[5]>0)   mFlag |= 32;
    if(mT[4]>0)   mFlag |= 16;
    if(mT[3]>0)   mFlag |= 8;
    if(mT[2]>0)   mFlag |= 4;
    if(mT[1]>0)   mFlag |= 2;
    if(mT[0]>0)   mFlag |= 1;
```

＜実行結果＞

```
mFlag=170;  //  ２進数で（1010 1010）
          =128 + 32 + 8 + 2
```

以上のようにすると、switch(mFlag)で case 0 ～case 255 の場合分けができる。また、それに加えて図３- f のような面の中心におけるピクセル値による場合分けが４４８通り存在する。

４.４　切り口におけるポリゴンの描画

DeltaViewer は２次元断面画像から範囲を指定して３D 構築するのだが、例えば図３- ｉのようにラグビーボール型のものを直方体で範囲を指定し３D 構築すると楕円形の切り口ができるが、この切り口は４.１.１のようなマーチングキューブズ法だけでは描画できないので、切り口を描画する新たなアルゴリズムが必要となる。

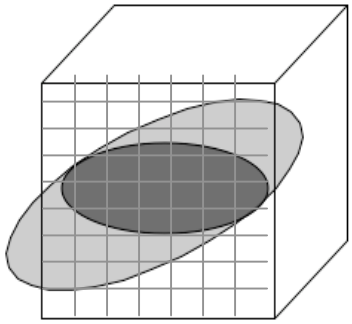

図３- i

アルゴリズムだが、切り口を描画する前にマーチングキューブズ法によって３次元データが立方体に分割してあるので、すでに切り口である面も図３- ｉのように格子状に分割されており、その頂点におけるピクセル値もわかっている。よって、切り口はこれを用いてポリゴンを描画する。また､切り口は全て平面で同じ方向であるので法線ベクトルの平均化は必要ない。４.２のように面に境界を引き、境界となるピクセル値より大きいところにポリゴンを描画する。図３- ｊのグレーの部分が描画するポリゴンである。ポリゴンの形の判別は前章と同様に、面の４頂点におけるピクセル値が境界値より大きいか小さいか判別してフラッグをたてて行う。

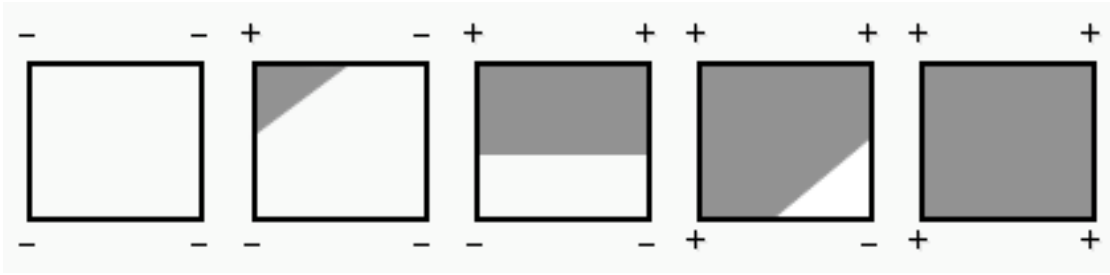

図３- ｊ

また、４頂点のピクセル値のうち向かい合う２つの頂点が「＋」で残りの２つが「－」であった場合、境界を２通り引くことができるので面の中心におけるピクセル値が境界値よりも大きいか小さいかで場合分けし、図３- ｋのようにグレーの部分を描画するポリゴンとする。

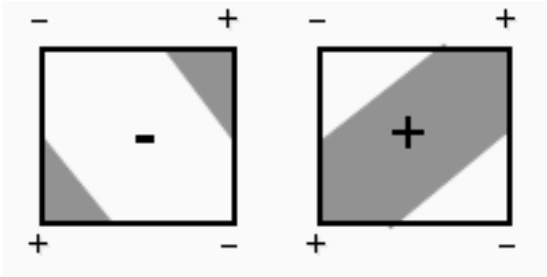

図 3 - k

３Ｄ 構築する範囲の指定は直方体となるので、切り口は最大６面存在する。よって、それぞれの面に対して場合わけすることが必要となる。

５．　法線ベクトル

この章では、本研究で重要となる法線ベクトルの平均化について詳しく述べる。本研究はマーチングキューブズ法よりポリゴンの頂点座標を求め、そのポリゴンにおける法線ベクトルを計算し、１つの頂点に複数存在する法線ベクトルを平均して１つにすることによって、面と面の境界が滑らかに見えるようにするため、法線ベクトルの平均化はとても重要なのである。以下に法線ベクトルの算出方法、法線ベクトルの平均化について述べる。

５.１　法線ベクトルの算出方法

DeltaViewer で扱う面は数学的な記述が判明しているので解析面における法線の算出方法を用いて求める。解析面とは数学的な方程式（または方程式群）により記述される、滑らかで境界の明確な面のことである。

DeltaViewer で扱うポリゴンはすべて３角形にしているので、３角形の面における法線ベクトル算出方法を以下に記述する。ポリゴンの３頂点をそれぞれ A、B、C とし、座標を A（Xa、Ya、Za）、B（Xb、Yb、Zb）、C（Xc、Yc、Zc）とする。ここで重要なのは、頂点 A、B、C の順番は右周りに統一して指定するということである。（左周りなら左回りで統一）３角形の法線ベクトルは２つ辺のベクトルの外積で求められるので、まず２つの辺のベクトル V、W を求める。図　４のように辺のベクトルを V＝（Xv、Yv、Zv）、W＝（Xw、Yw、Zw）とすると、

V（Xv、Yv、Zv）　＝ B（Xb、Yb、Zb）－ A（Xa、Ya、Za）
W（Xw、Yw、Zw）＝ C（Xc、Yc、Zc）－ A（Xa、Ya、Za）

のように求めることができる。法線ベクトルは２つ辺のベクトルの外積で求められるので、外積（V×W）は、

V×W　＝［Xv、Yv、Zv］×［Xw、Yw、Zw］
　　＝［(YvZw－YwZv)　(ZvXw－ZwXv)　(XvYw－XwYv)］

となる。その後、法線ベクトルを長さで除算して正規化（ベクトルの長さを１に

する）すると、このポリゴンにおける法線ベクトルが求められる。頂点 A、B、C はすべて同じ平面上にあり同じ向きであるためこの法線ベクトルがそのまま頂点 A、B、C に指定する法線ベクトルとなる。

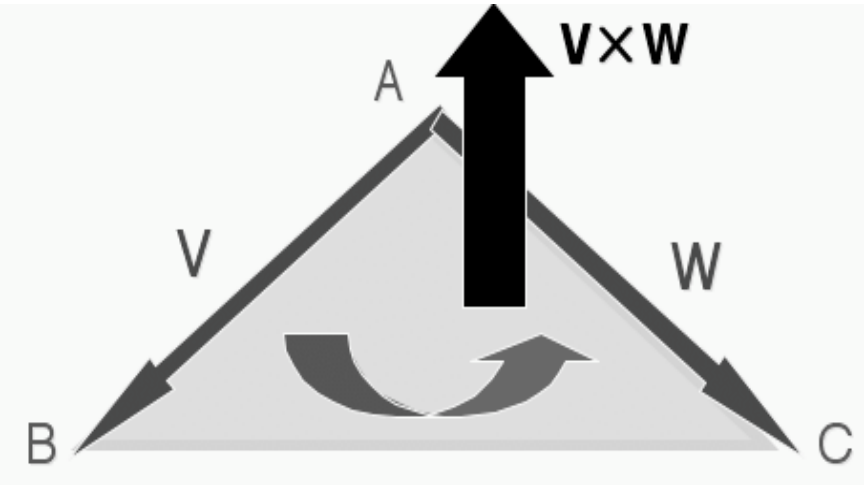

図　4

図　4 のポリゴンの中に V から W 方向を指す矢印があるが、これは外積を求めるときに V×W とすると、矢印の方向に回転する右ネジが法線ベクトルの向きになることを示している。そして、矢印の向きを反対にすると W×V となり、法線ベクトルが図 4 と反対方向に向く。ここでは頂点 A、B、C の順番はを右周りに統一していたので右ネジの法則となる。このような理由で「３．　OpenGL におけるポリゴンの描画」でも述べたように、法線ベクトルの向きを決めるため頂点の順番を指定することはとても重要なのである。

５.２　法線ベクトルの平均化

OpenGL では、ポリゴンの各頂点に指定された法線ベクトルによって光源から受ける光の量が計算され、光が当たっているポリゴンは明るいカラーで、当たっていないポリゴンは影ができ暗いカラーで描画される。DeltaViewer における従来の３Ｄ構築アルゴリズムでは、１つのポリゴンにおけるすべての頂点に指定している法線ベクトルは、図５- ａ のように同じ向きであった。すると、照光処理をしたとき隣り合うポリゴンごとに影ができてカラーが異なってくるため、境界線がくっきりと見えていたわけである。ここで、注目したのが１つの頂点は複

数のポリゴンの頂点を重複しているため、複数の法線が指定されているということである。

本研究では、１つの頂点に複数存在した法線ベクトルを平均して、図５- ｂ のように１つの頂点につき１つの法線ベクトルを指定するようにした。この作業のことを本研究では法線ベクトルの平均化ということにしている。こうすることで、照光処理をしたときに隣り合う２つのポリゴンに共通する１辺に当たる光の量、向きが同じになるので、境界線を滑らかに見ることができるようになる。

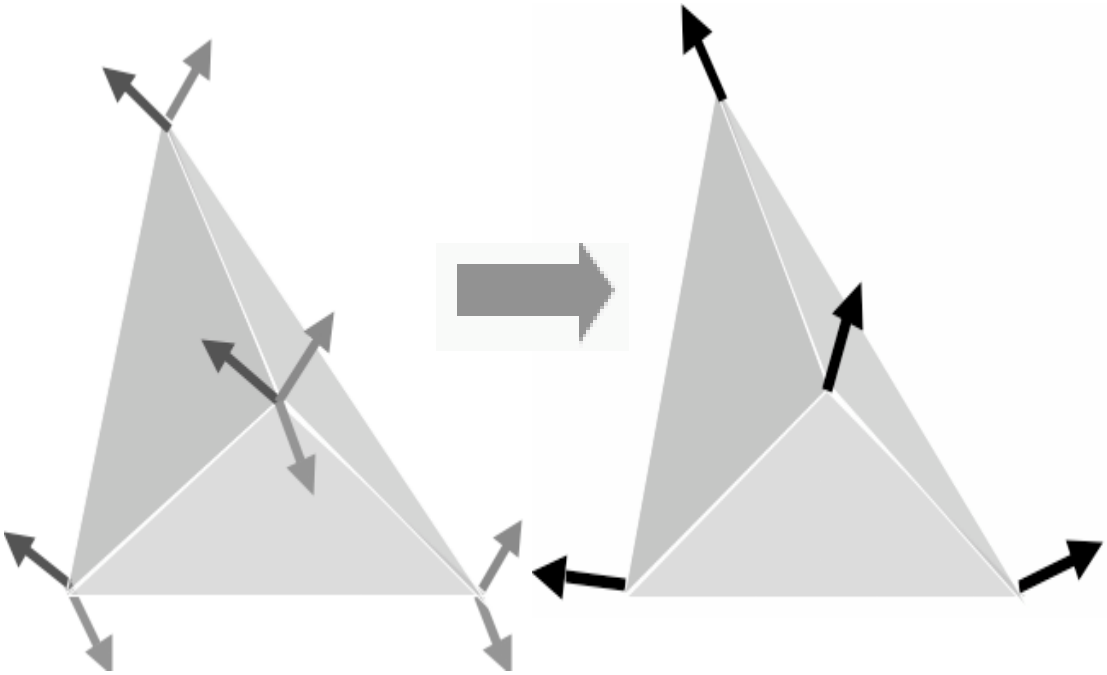

図５- ａ：法線ベクトルの平均化前

図５- ｂ：法線ベクトルの平均化後

５.３　法線ベクトルの平均化処理前、処理後の比較

法線ベクトルの平均化を行う前と行った後でどのように違って見えるのかを比較する。

コンピュータ画面上で球を描画するとき曲線を用いたいのだが、曲面を描画することはできない。ゆえに OpenGL で球体を描画する場合は、小さな３角形のポリゴンをいくつも張り合わせて描画して球に近付けていく。ここでは、８０面体を描画した実行結果で比較する。

法線ベクトルの平均化処理をする前のものが図６- ａ で、ポリゴンとポリゴンの境界がはっきり見えている。法線ベクトルの平均化処理を行ったものが図６- ｂで、ポリゴンとポリゴンの境界が滑らかになり球に近くなったことがわかる。しかし、法線ベクトルの平均化によって８０面体自体の形が変わったのではない。「５.２　法線ベクトルの平均化」でも述べたように、法線ベクトルが１頂点につき１つになったことで照光処理をしたときに隣り合うポリゴンに共通する１辺に当たる光の量、向きが同じになる。ゆえにポリゴンの繋ぎ目で急激にカラーが変わるということを防止でき、境界線を滑らかに見えることができるのである。

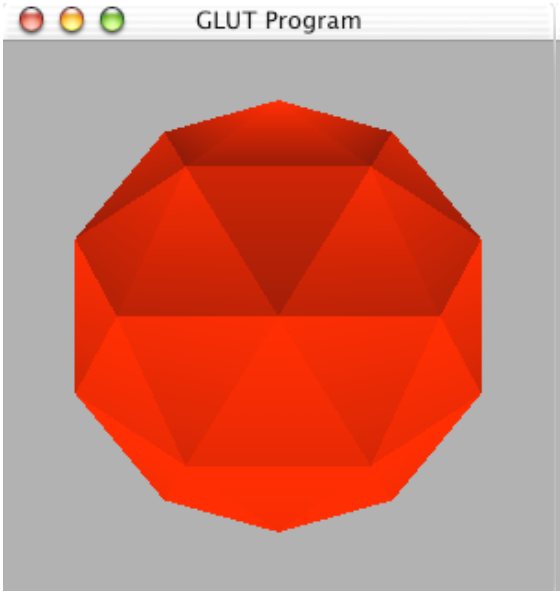

図６- a

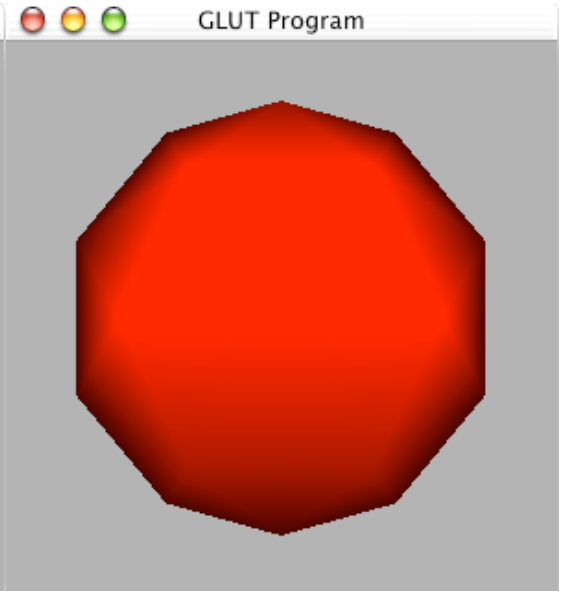

図６- b

６．　従来の DeltaViewer の実行結果と本研究での実行結果の比較

従来の３Ｄ構築アルゴリズムによる DeltaViewer の実行結果と本研究での実行結果を比較する。

図７- a は従来の３Ｄ構築アルゴリズムによる DeltaViewer での実行結果である。ポリゴンとポリゴンの境界がはっきりと見えている。そして図７- b は法線ベクトルの平均化処理を加えた本研究での実行結果である。ポリゴンとポリゴンの境界を滑らか見えるようにするという当初の目的が達成されたことがわかる。

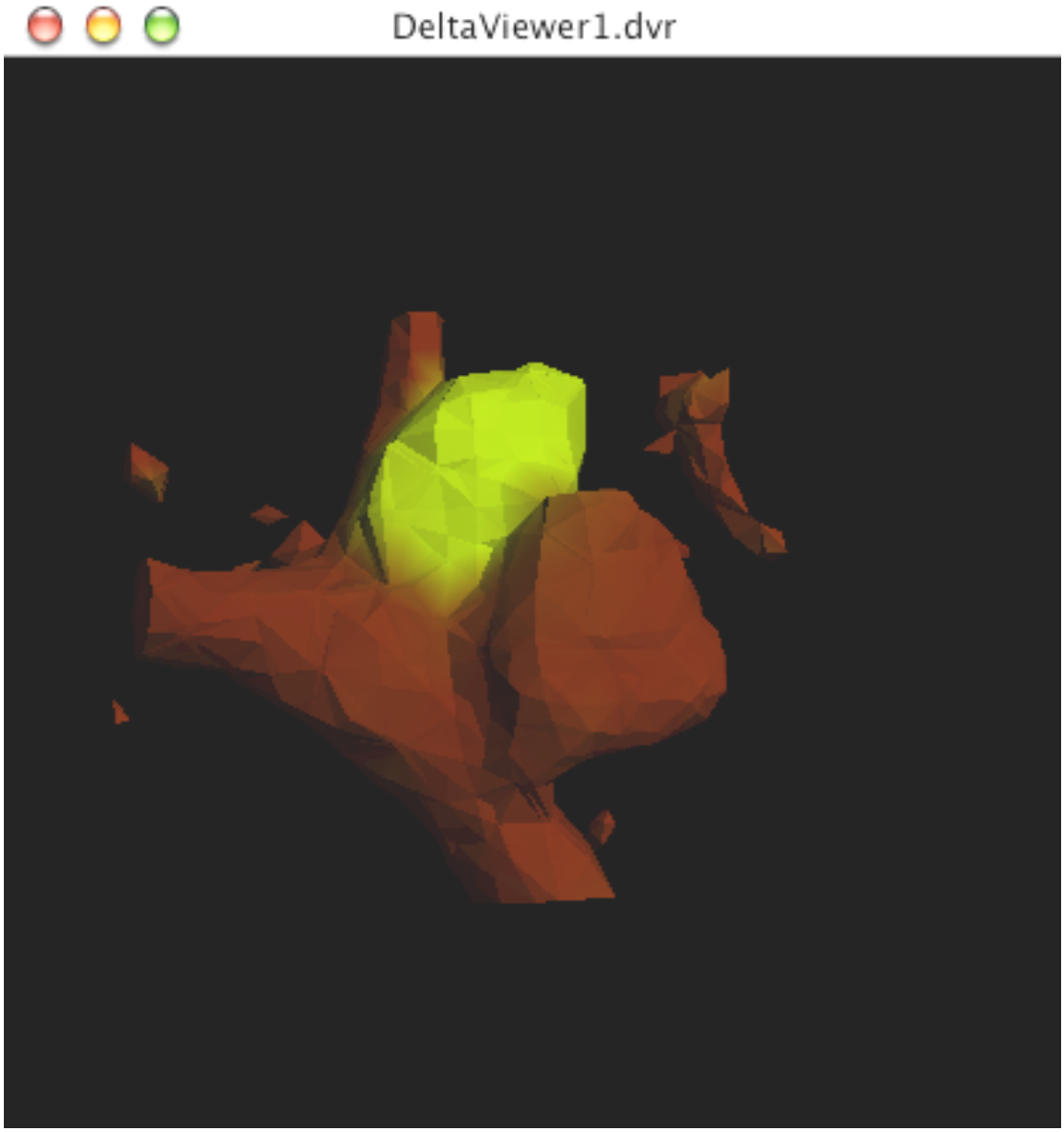

図７- a：従来の３Ｄ構築アルゴリズムによる DeltaViewer の実行結果

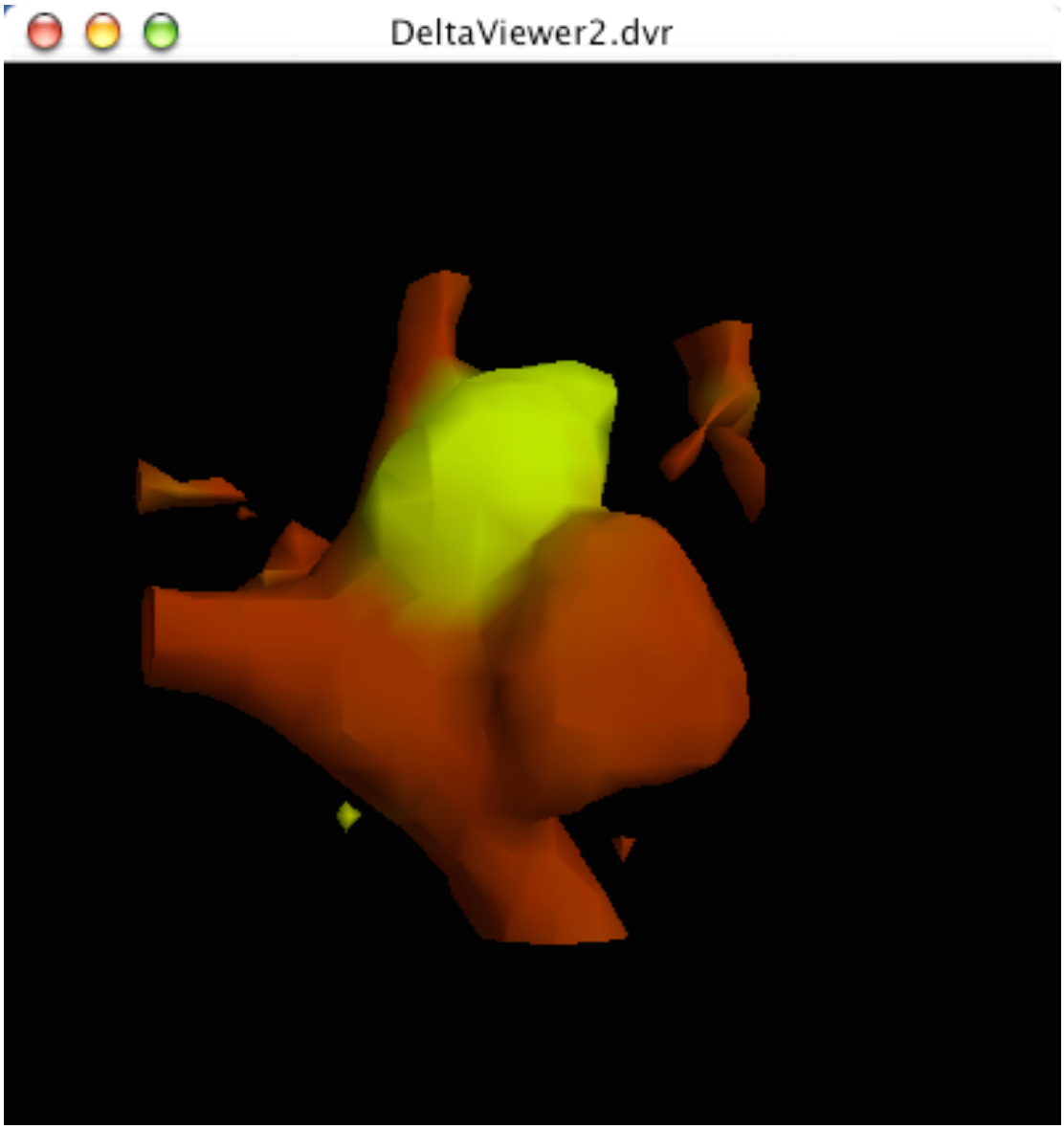

図 7 - b：本研究での実行結果

７．結果・考察

当初の目的であったポリゴン間の境界を滑らかにすることができた。法線ベクトルの平均化により実現したものである。

また、描画されるまでの時間を短縮することができた。マーチングキューブズ法で扱う立方体の大きさを大きくしても、法線ベクトルの平均化実現により従来のものとほぼ同じ精度で描画できるようになったためである。

８．参考文献

参考文献

[1]　OpenGL Architecture Review Board　編：OpenGL　プログラミングガイド、ピアソン・エデュケーション（2001）

[2]　姜秀子：修士論文「DeltaViewer プロジェクト」（2001）

参考 URL；

[3]「URL」DeltaViewer Projekt

http://vivaldi.ics.nara-wu.ac.jp/~wada/DeltaViewer/index-j.html

謝辞

指導教官である和田先生にはほんとにたくさんご指導頂きました。卒業論文やゼミ、プログラム…そして発表の練習におけるご指導のお陰で１位をとることができました。本当にありがとうございました。和田研究室の配属になれて良かったと思います。あと２年間大学院でのご指導もよろしくお願い致します。

和田ゼミの伊佐友江さん、許潤玉さん、細井絵里子さんにもほんとにお世話になりありがとうございました。みんなのお陰で楽しくゼミ生活（？）を送ることができました。みんなは東京で就職なのでこれからは私一人になって寂しいですが、また遊びにきてくださいね。