2001年　修士論文

DeltaViewer プロジェクト

奈良女子大学大学院　人間文化研究科　情報科学専攻
姜　秀子

目次

イントロダクション

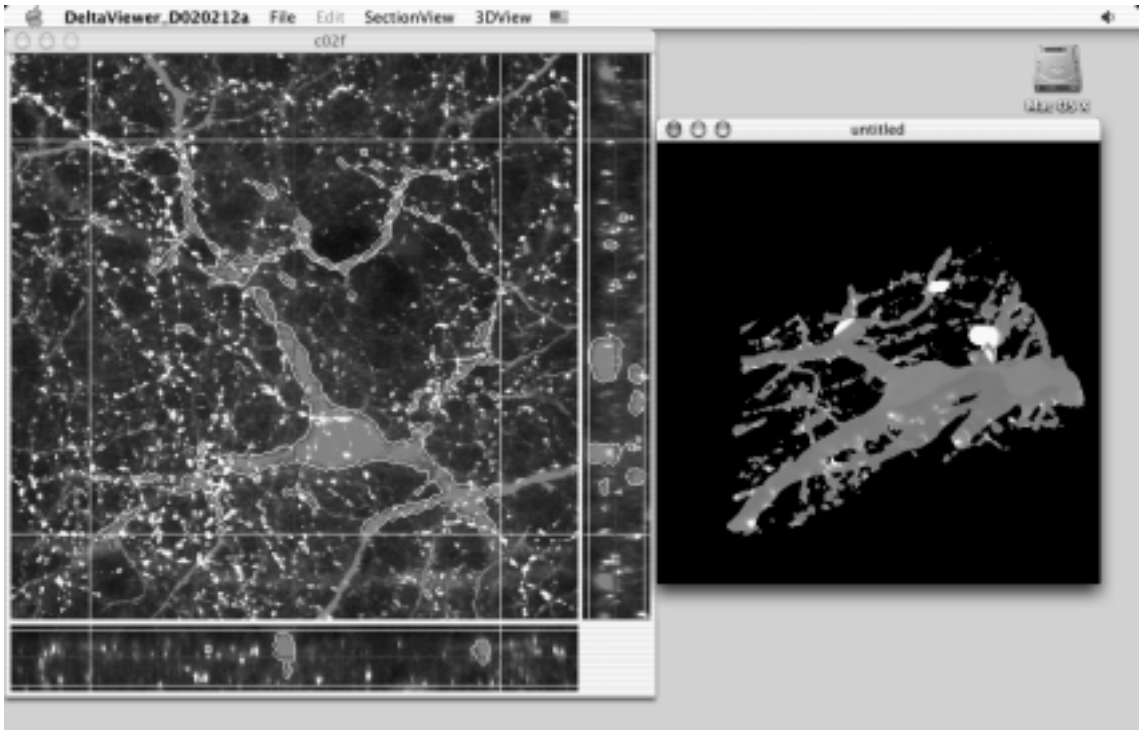
DeltaViewer は，レーザー共焦点顕微鏡によって得られる二次元断面画像から任意の境界を抽出して曲面データを三次元構築し，それを Open GL（[1],[2],[3]）を用いてコンピュータ画面上に表示するソフトウェアである．DeltaViewer の開発は，和田昌昭教授および奈良女子大学生物科学科の保研究室との共同研究である．

保研究室では，光神経内分泌器官に関する生理学的および組織学的研究において，Leica 製のレーザー共焦点顕微鏡([URL1])と菱化システム株式会社の三次元画像の構築・解析を行うソフトウェアである VoxelView([URL2])を用いている．研究方法は，研究対象の試料に蛍光物質を注入し，レーザー光の照射により蛍光物質を発色させ，その蛍光発色を観察する．レーザー共焦点顕微鏡では，非焦点の光を排除することにより，焦点の合った画像のみを取り出すことができる．そのため，立体的な試料の一定の厚みの部分だけを観察することができる．レーザー共焦点顕微鏡で取り出した画像は，付属のソフトを用いてファイル出力できる．そのようにして得られた二次元断面画像から VoxelView を用いて，境界抽出を行い，三次元構築している．現状での問題点をあげる．

① VoxelView は，シリコン・グラフィックス社製のワークステーション上でのみ稼動する．
② ソフトウェアもワークステーションも高価である．
③ VoxelView を用いて三次元構築された立体データを回転させる度に描画しなおすため，立体データをスムーズに動かすことができない．

DeltaViewer は，Macintosh 上で動作するフリーソフトウェアとして製作している．また，三次元構築された立体データを任意に動かすことができるので，物体の空間的配置をより直感的に把握することができる．さらに，三次元構築するときの方法として，VoxelView ではボリュームレンダリングを用いているが，DeltaViewer では補間と境界抽出に新しいアルゴリズムを組み合わせた方法を用いた．

本論文では，主に画像データファイルの読み込みと三次元構築に関して詳しく述べる．


DeltaViewer_D020212a
File
Edit
SectionView
3DView
c02f
untitled

1. 読み込み処理

1.1. DeltaViewer で読み込み可能なファイル形式

生物学の分野で使われているいくつかのレーザー共焦点顕微鏡の保存可能なファイル形式の一覧を表１にまとめる.

表１：レーザー共焦点顕微鏡の保存可能なファイル形式

	Bio-Rad	Olympus	Zeiss	Leica	Nikon
会社独自	有り	有り	有り	有り	有り
TIFF	○	○	○	○	○
BMP (Windows Bitmap)	○	○	○	×	×
JPG	○	×	○	×	×
GIF	○	×	○	×	×
PSD (Adobe Photoshop)	×	×	○	×	×
PCT (Macintosh QuickDraw)	×	×	○	×	×

各社の顕微鏡([URL3],[URL4],[URL5],[URL6])によるファイル保存形式から三次元構築のソフトウェアが TIFF ファイル([URL7],[5])に対応していれば，画像データを読み込む前にファイル形式を変換せずに取り扱えることが分かる．そのため，DeltaViewer のファイル形式は TIFF ファイルに対応させている．また TIFF のカラー形式は，RGB Full-color 形式と Palette-color 形式のものが読み込めるようになっている．TIFF ファイルの構造については[A]を参照.

レーザー共焦点顕微鏡は，物体の任意の深さから断面画像を取り出すことができる．立体画像データを取り出すには，顕微鏡に付属の簡易なソフトを用いて，指定した深さから一定間隔で深さを順に変えて，その断面画像を出力するようになっている．その際，深さを変える一定の間隔は，ユーザが指定できる.

1.2. 複数ファイルの読み込み指定方法

画像データファイルを一枚読み込むと，平面断面図の情報を得る．画像データファイルを深さ方向に複数枚読み込むことで，平面断面図が連なり三次元データとなる．

DeltaViewer における複数ファイルの読み込みに関しては，ファイル名を“abcd001.TIF”などのように先頭部分（abcd），通し番号（001），拡張子（.TIF）の三つのパートからなるものと仮定している．一枚目の画像を読み込むと先頭部分，通し番号の桁数，拡張子は決定し，abcd001.TIF, abcd002.TIF,abcd003.TIF,… のように昇順で通し番号の最後まで読み込んでいく．先頭部分の長さや通し番号の桁数に特別な制限はない．

上で述べた深さ方向の間隔によって，出力される画像データファイルの枚数が変わる．このときの縦，横および深さの比は，DeltaViewer において用いるアルゴリズムとしては問題ない．しかし，現状の DeltaViewer の仕様としてはピクセル単位で１：１：１であると仮定している．

1.3. ファイル読み込みの具体的方法

DeltaViewer では，読み込む画像ファイルとして次の二通りを想定している．

① 赤，緑，青の三色の色データを同一ファイルに書いているもの

② 赤，緑，青の各コンポーネントでファイルを分けているもの

①の場合，ファイルごとに三色の色データが書かれているので，File コマンドから“Open TIFF(RGB)”を選択し，読み込む先頭ファイルを指定する．

②の場合，赤，緑，青の三色のコンポーネントごとにファイルが別になっている．たとえば，色データが赤色の画像から読み込みはじめるとすれば，まず，SectionView コマンドから“Open TIFF(Red)”を選び，読み込む先頭ファイルを指定する．続けて，緑色，青色の画像を読み込むときは，それぞれ“Open TIFF(Green)”,“Open TIFF(Blue)”を SectionView コマンドから選択し，それぞれの先頭ファイルを指定する．

尚，DeltaViewer では，ライカ製の TCS NT を用い TIFF 形式で出力した画像データファイルで動作確認済みである．

1.4. 読み込みの内部処理

画像データファイルを読み込み描画するためには，画像データファイルの幅，高さ，ピクセル値などが必要である．これらの情報をファイルから読み込む方法をカラー形式ごとに示す．

RGB Full-color 形式の場合は，まず IFD から描画するために必要な情報である幅，高さ，カラー形式，色データの先頭ポインタを調べる．幅と高さから，赤，緑，青の濃淡値で表しているピクセル値を入れるための配列を作る．色データからピクセル値を順にコンポーネントごとの配列に格納していく．

Palette-color 形式の場合も，RGB Full-color 形式の場合と同様にまず IFD から描画するために必要な情報である幅，高さ，カラー形式，色データの先頭ポインタを調べる．次にピクセル値を入れるための配列を作るにあたって，先にカラーマップを作成しなければならない．その後，色データからインデックスを調べ，そのインデックスとカラーマップから求められたピクセル値を色の配列に格納していく．

2. 二次元画像の描画

画像データファイルを読み込み終わると，１ピクセルずつ順に描画していく．しかし，１ピクセルずつ描画すると画面全体が描画されるまでにとても時間がかかり，描画が終了するまで画像の全体像を把握できない．そのため，まず全体を粗く描き，どんどん細かく描いていくように工夫してある．

3. 二次元画像の境界線

3.1. 境界線の定義

地図上で見られる等高線とは，同じ高さの地点を連ねて出来た線である．この同じ高さを知るには，高さを表す尺度とその基準となる高度（基準値）が必要である．

同様に画像上の各点に対して色の明るさを表す値（明暗値）を指定できれば，等高線すなわち境界線を引くことができる．色の大きさを決めるのは大変難しいため，便宜上，明暗値が大きいと明るい，明暗値が小さいと暗いと表現する．

以下に明暗値を用いた境界線の引き方と，その明暗値の算術方法について詳しく述べる．

3.2.　明暗値と境界線

画像データファイルを読み込むと，各点の座標とその点でのピクセル値が分かる．3.1 で述べたように，境界線を引くには各点での座標とピクセル値以外に二つの値が必要である．一つ目は，境界線を引くときの色を表す境界色．二つ目は，各点におけるピクセル値が境界色と比べてどれだけ明るいか（もしくは暗いか）を表す明暗値である．この境界色と明暗値を求めるときの工夫点について示す．

もし，Grayscale 画像のようにピクセル値が白色から黒色への濃淡値で表されているならば，境界色としてある任意の濃淡値を指定すると，画像上にその境界色と同じ色の点を連ねて，境界線を引くことができる．

しかし，ピクセル値が赤色，緑色，青色の三色で表されている画像の場合，三色すべての濃淡値が境界色のそれぞれのコンポーネントの濃淡値と同じかどうかを判定しなければならない．そこで，DeltaViewer では，赤色，緑色，青色の三次元の濃淡値を一次元の明暗値で表すアルゴリズムを用いた．この明暗値を求めるアルゴリズムについては，3.4 にて詳しく説明する．

境界色と同じ明暗値の点を求めるには，任意の二点を結ぶ線分と境界線との交点を求めればよい．この計算方法について説明する．

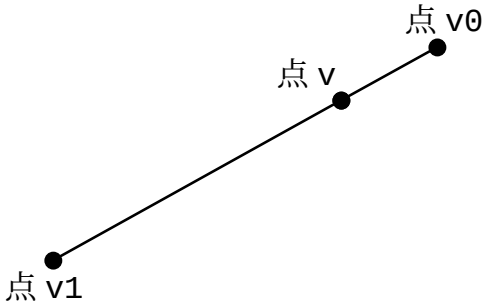


図 3.2

図 3.2 のように二点を v0，v1 とし，それぞれの点における明暗値を t0，t1 とする．また，二点を結ぶ線分と境界線の交点を v，点 v での明暗値を t とする．ただし，境界色での明暗値 t は分かっているものとする．このとき，v は次式にて求めることができる．

```
s = (t – t0) / (t1 – t0);
v = (1.0 – s) * v0 + s * v1;
```

以上のように，明暗値をもとに二点を結ぶ線分と境界線との交点を求め順につないでいくことで，二次元平面上に境界線を引くことができる．しかし，この明暗値は，色の大きさの基準となる境界色が分からなければ求めることができない．次に，この境界色の求め方を示す．

3.3. 境界色と境界線

DeltaViewer では，任意の一点を選択し，その点におけるピクセル値を境界色と決めるのではなく，次のような操作をしている．

画像上で，任意の点を指定する．このとき選んだ点のピクセル値が境界色よりも明るいか暗いかを決める．このように，境界色よりも明るいか暗いかを決めながら，有限個の点をとる．境界色よりも明るい点と暗い点が，それぞれ一組以上あれば，境界色を求めることができる．この境界色を求める方法について次に示す．

（r, g, b, t）で一組のサンプルを取る．r, g, b は任意の点におけるピクセル値の各コンポーネントの濃淡値を表し，t はそのピクセル値が境界色より明るいか暗いかを表す値である．境界色より明るければ t=+1，暗ければ t=-1 とする．このようなサンプルを有限個とったサンプルリストから境界色を求める．ここでは，t=+1 と t=-1 のサンプルが一組ずつ存在する場合について説明する．尚，赤色，緑色，青色の三つの濃淡値を表す三次元座標を色キューブと呼ぶことにする．

サンプルリストの二点のピクセル値を色キューブ上にとる．色キューブ上において，この二点を結ぶ線分の垂直二等分面上に境界色が存在する．つまり，t=0 を満たす点が境界色となる．

次にどのように境界線を引くかについて述べる．まず，DeltaViewer におけるアルゴリズムを用いて境界線を引くには，画像を三角形に分割し，その各々の三角形において境界線との交点を調べなければならない．DeltaViewer では，境界線を引くために平面画像を四角形に分割した上で，各四角形を三角形に細分している．そして，分割された三角形において，境界線と交わるかを調べている．分割方法については，3.5 にて詳しく述べる．

分割された三角形において有限個のサンプルリストをもとに次の操作を行う．

① 境界色を求める

上で述べたようにサンプルリストから求める．

境界色より明るいサンプルと暗いサンプルがそれぞれ一つ以上必要である．

② 三角形の各頂点の明暗値を求める

サンプルリストと 3.4 にて説明するアルゴリズムを用いて求める.

③ 三頂点の明暗値の符号によって，場合分けをする

⑴すべて同一符号の場合（図 3.3(a)）

境界線との交点が存在しない.

明暗値が正ならば，すべての点が境界色よりも明るい.

明暗値が負ならば，すべての点が境界色よりも暗い.

⑵異符号が存在する場合（図 3.3(b)）

境界線との交点が存在する.

三頂点のうち二点ずつ明暗値を比べる．明暗値の符号が異なる二点間に境界線との交点が存在する.

図 3.3 において，破線は境界線を表す.

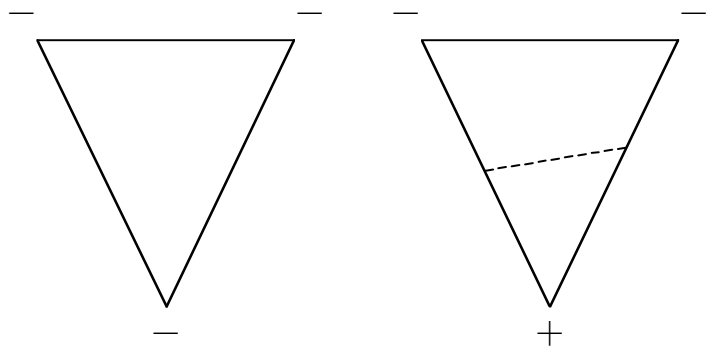


図 3.3

尚，サンプルリストにより境界色が決まるため，リスト数が増減するたびに上記の操作を行う.

3.4. 明暗値を求めるアルゴリズム

境界抽出法として閾値法・エッジ追跡法・領域拡張法などが使われているが，DeltaViewer では，和田昌昭教授による補間アルゴリズム([4])を使用している．ここでは，そのアルゴリズムを三次元に拡張して具体的に示す.

独立した三つの変数 r,g,b と一つの変数 t のデータを次のようにおく.

$$(r_i,\ g_i,\ b_i,\ t_i) \qquad (i = 1,\ldots,n)$$

α,β,γ をゼロでない数とする．任意の点（r,g,b）が与えられたとき，補間

した値 t を求めるために三次元において

$$V = V(R, G, B) = a(R-r) + b(G-g) + c(B-b) + d \quad (1)$$

とし，(1)に次式の重みをかけ最小二乗法を用いて

$$W_{r,g,b}(R, G, B) = \exp\left(- \frac{(R-r)^2}{\alpha^2} - \frac{(G-g)^2}{\beta^2} - \frac{(B-b)^2}{\gamma^2} \right)$$

次式の誤差 E が最小となるように変数 a,b,c を求める.

$$E = \sum_{i=1}^{n} W_{r,g,b}(r_i, g_i, b_i)\,(V(r_i, g_i, b_i) - t_i)^2$$

次式を解くことでこれらの変数を求められる.

$$\frac{\delta E}{\delta a} = \frac{\delta E}{\delta b} = \frac{\delta E}{\delta c} = \frac{\delta E}{\delta d} \qquad (2)$$

(1),(2)より

$$\begin{pmatrix} S_{rr} & S_{rg} & S_{rb} & S_r \\ S_{rg} & S_{gg} & S_{gb} & S_g \\ S_{rb} & S_{gb} & S_{bb} & S_b \\ S_r & S_g & S_b & S_1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} S_{rt} \\ S_{gt} \\ S_{bt} \\ S_t \end{pmatrix}$$

それぞれの成分は,

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)(r_i - r)^2,$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)(r_i - r)(g_i - g),$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)(r_i - r)(b_i - b),$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)(g_i - g)^2,$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)(g_i - g)(b_i - b),$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)(b_i - b)^2,$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)(\mathrm{r}_i - r),$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)(\mathrm{g}_i - g),$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)(\mathrm{b}_i - b),$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i),$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)(r_i - r)t_i,$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)(g_i - g)t_i,$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)(b_i - b)t_i,$$

$$S_{rr} = \sum_{i=1}^{n} W_{r,g,b}(r_i,g_i,b_i)t_i,$$

クラメールの公式を用いて

$$t = \mathrm{d} = \frac{\mathrm{d1}*\mathrm{S_{rt}} + \mathrm{d2}*\mathrm{S_{gt}} + \mathrm{d3}*\mathrm{S_{bt}} + \mathrm{d4}*\mathrm{S_t}}{\mathrm{d1}*\mathrm{S_r} + \mathrm{d2}*\mathrm{S_g} + \mathrm{d3}*\mathrm{S_b} + \mathrm{d4}*\mathrm{S_1}}$$

ただし

$$d1 = \det\begin{pmatrix} \mathrm{S_{rg}} & \mathrm{S_{rb}} & \mathrm{S_r} \\ \mathrm{S_{gg}} & \mathrm{S_{gb}} & \mathrm{S_g} \\ \mathrm{S_{gb}} & \mathrm{S_{bb}} & \mathrm{S_b} \end{pmatrix} \qquad d2 = \det\begin{pmatrix} \mathrm{S_{rr}} & \mathrm{S_{rb}} & \mathrm{S_r} \\ \mathrm{S_{rg}} & \mathrm{S_{gb}} & \mathrm{S_g} \\ \mathrm{S_{rb}} & \mathrm{S_{bb}} & \mathrm{S_b} \end{pmatrix}$$

$$d3 = \det\begin{pmatrix} \mathrm{S_{rr}} & \mathrm{S_{rg}} & \mathrm{S_r} \\ \mathrm{S_{rg}} & \mathrm{S_{gg}} & \mathrm{S_g} \\ \mathrm{S_{rb}} & \mathrm{S_{gb}} & \mathrm{S_b} \end{pmatrix} \qquad d4 = \det\begin{pmatrix} \mathrm{S_{rr}} & \mathrm{S_{rg}} & \mathrm{S_{rb}} \\ \mathrm{S_{rg}} & \mathrm{S_{gg}} & \mathrm{S_{gb}} \\ \mathrm{S_{rb}} & \mathrm{S_{gb}} & \mathrm{S_{bb}} \end{pmatrix}$$

サンプルリストにおけるサンプル数が３以下の場合，上で示したアルゴリズムを用いても結果が不定となり明暗値が求まらない．そこで，サンプル数が３以下の場合は特別な処理をしている．まず，サンプル数が２の場合について説明する．

点 v0（R[0], G[0], B[0], T[0]）と点 v1（R[1], G[1], B[1], T[1]）は，独立なサンプルとする．このとき，任意の点 v の明暗値を求める．

① 色キューブ上で二点の色ベクトルを求める

```
r01 = R[1] – R[0];
g01 = G[1] – G[0];
```

```
b01 = B[1] – B[0];
```

② 点v（r,g,b）と①で求めた色ベクトルとの内積を求める

```
num = r01 * (r – R[0]) + g01 * (g – G[0]) + b01 * (b – B[0]);
```

③ ①で求めた色ベクトルの大きさの二乗を求める

```
den = r01 * r01 + g01 * g01 + b01 * b01;
```

④ 点v における線形補間値 t を求める

```
t = num / den;
```

①～④の計算方法からも分かるように，点 v から点 v0 と点 v1 を通る直線に直交射影したのち，その直線上で線形補間値 t を求めている．

同様に，サンプル数が３の場合は，任意の点 v から点 v0，点 v1，点 v2 の三点を通る平面に直交射影したのち，その平面内で線形補間値 t を求める．

3.5. 二次元平面での分割法

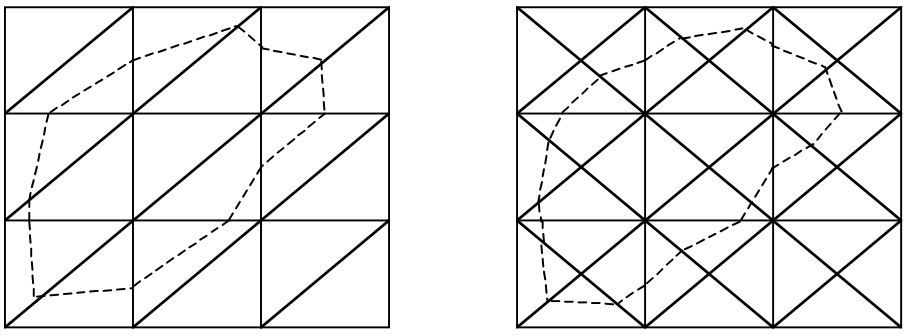
前述のように境界線を引くための平面画像の分割方法について説明する．

DeltaViewer では，まず平面画像をできるだけ正方形に近い格子状に分割している．そして，各格子において対角線を引き，三角形に分割する．その後，各三角形において境界線と交わるかどうかを調べ，境界線を引くことになる．この各格子における三角形の分割方法として，次の二通りを考察した．

① 三角形二分割法（図3.5(a)）
　対角線を一本だけ引いた場合である．

② 三角形四分割法（図3.5(b)）
　対角線を二本引いた場合である．

図 3.5 において，破線は境界線を表す．



(a)　(b)

図 3.5

図 3.5 で比べると，(a)よりも(b)での境界線がよりなめらかに描かれているのが分かる．また，それぞれの分割方法で複数ファイルに境界線と引き三次元データを取った．すると，三角形二分割法で得られる三次元のデータに問題が生じた．そのため，DeltaViewer では(b)の三角形四分割法を用いている．この問題点については，4.1 において詳しく述べる．

次に，平面画像を格子に分割するときの内部処理について説明する．DeltaViewer では，格子に分割するときの分割数を指定できる．平面画像の幅と高さのうち大きい方を指定した分割数で等分割する．このとき，幅と高さの小さい方は一般的にはその等分割された幅の整数倍にはならないので，平面画像を正方形に分割できるとは限らない．そのため，DeltaViewer では次のような処理を行って平面画像をなるべく正方形に近い長方形に分割するようにしている．平面画像の幅を mWidth，高さを mHeight，分割数を mNumDiv とする．

① 平面画像の幅と高さの長い方を（wmax）求める

```
wmax = mWidth;
if (mHeight > wmax)
   wmax = mHeight;
```

② wmax を分割数で割った値を（div_size）求める

```
div_size = wmax / mNumDiv;
```

③ 幅と高さを div_size で割り，新しい分割数を（mNumDivW，mNumDivH）求める

```
mNumDivW = lround(mWidth / div_size);
mNumDivH = lround(mHeight / div_size);
```

④ さらに幅と高さを③で求めた分割数で割った値を（mDivSizeW，mDivSizeH）求める

```
mDivSizeW = mWidth / mNumDivW;
mDivSizeH = mHeight / mNumDivH;
```

一枚の平面画像が 500×410（ピクセル），分割数が 30 であった場合について具体的に計算すると次のようになる．尚，小数点第三位までの表示とする．まず wmax は 500.000, div_size は 16.666 になる. すると, DivW は 30.001, DivH は 24.601 となり，SizeW は 16.666，SizeH は 16.665 となる．この SizeW と SizeH の値が各格子の幅と高さとなる.

3.6. 分割法とピクセル値

上記の分割法を用いて境界色や明暗値を求めるときのピクセル値の求め方について説明する.

サンプルリストの作成，境界色と明暗値を求めるには，必ず各点におけるピクセル値が必要である．この各点とは，格子状に分割されたときの格子の頂点を指す．しかし，上で述べたように分割された各格子の幅と高さは，整数座標で表せないことがある．画像データファイルにおける色データは，整数座標におけるピクセル値である．したがって，DeltaViewer におけるピクセル値は任意の点の周辺の色合いも考慮した色にしている．尚，どの程度，周りの色合いを考慮するかは，ユーザが指定できる．この値を alpha とする．alpha=1 であれば, 上下左右に対して 2 ピクセルずつの色合いを考慮する. 任意の点（ii, jj）のピクセル値を求める方法を示す.

① 縦方向と横方向において色合いを近似する範囲を整数座標で求める

```
imin = lround(ii – 2.0 * alpha);
imax = lround(ii + 2.0 * alpha);
jmin = lround(jj – 2.0 * alpha);
jmax = lround(jj + 2.0 * alpha);
```

② 点（i, jj）との距離の二乗を（r2）求める

```
r2 = ((double)i – ii)² + ((double)j – jj)²;
```

③ 重みを（w）求める

```
w = exp(- r2 / (alpha * alpha));
```

④ 重みをかけたピクセル値を（c）求める

（点（i, j）におけるピクセル値を rgb（i, j）とする）

```
c = w * rgb(i,j);
```

⑤ ①で求めた範囲に（imin≦i≦imax，jmin≦j≦jmax）おいて，w と c は累積しながら②～④をくり返す

```
ww += w;
cc += c;
```

⑥ 近似したピクセル値を（color）求める

```
color = cc / ww;
```

4. 三次元構築

4.1. 三次元における分割法

二次元平面の画像を複数枚かさねることで，直方体の画像データができる．この直方体のデータを 3.5 で述べた分割方法に深さを加えて，できるだけ立方体に近い形に分割する．その後，各立方体を三角錐に分割し，三角錐ごとに境界面と交わるかを調べる．3.5 で二種類の三角形分割法を述べた．二次元平面における三角形分割方法により，三次元空間における三角錐分割方法も異なる．三角形分割法ごとに分けて，三角錐分割法を示す．

① 各立方体において 6 個の三角錐に分割した場合（図 4.11(a)）

立方体の上面と底面において三角形二分割法を用い，立方体を二つの三角柱に分ける．その各々の三角柱において次のように 3 個の三角錐に分割する．(v0, v1, v2, v2′), (v0, v1, v1′, v2′), (v0, v0′, v1′, v2′)
そして，分割された三角錐が境界面と交わるかを調べる．

② 各立方体において 24 個の三角錐に分割した場合（図 4.11(b)）

立方体の中心点と各頂点を結び，6 個の四角錐に分ける．その各々の四角錐の底面において三角形四分割法を用い，4 個の三角錐に分割する．そして，各三角錐が境界面と交わるかを調べる．

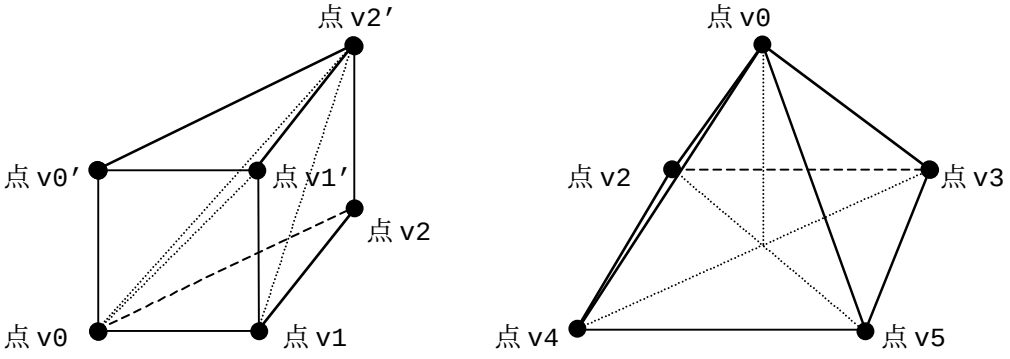
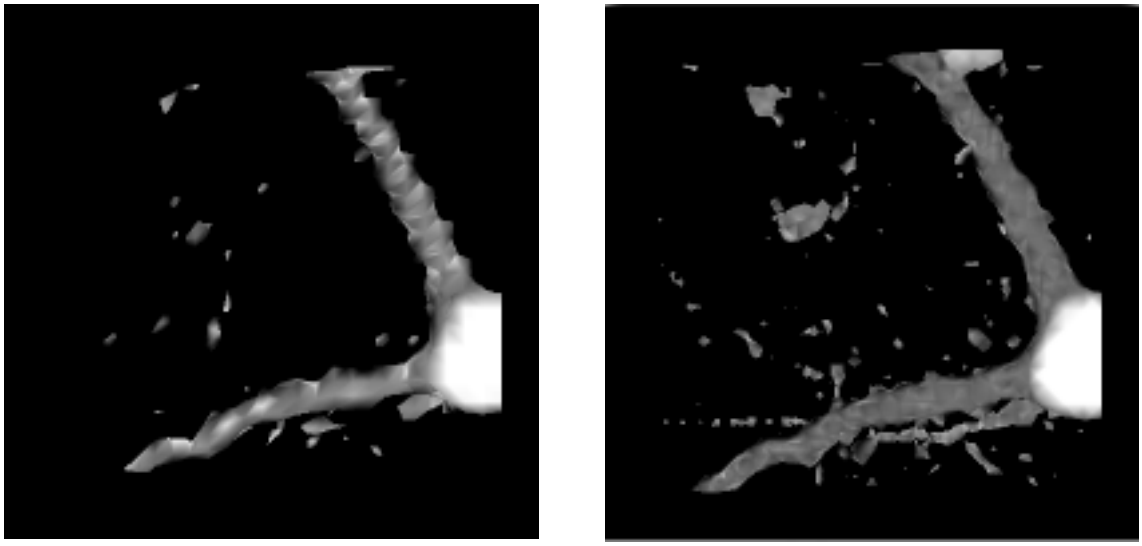


図 4.11

実際に上記の①，②の分割法を用い，一枚が 512*512（ピクセル）の画像データファイルを 60 枚読み込み，25 分割として，三次元構築したデータを図 4.12 に示す．



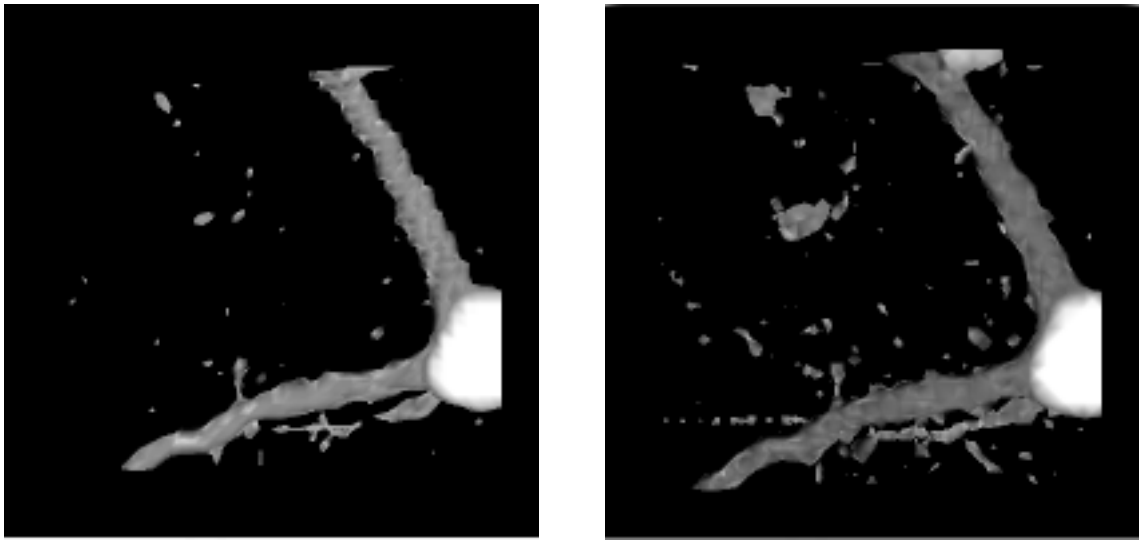
(a)　　(b)

図 4.12

それぞれの図において，右下にある細胞体からほぼ直角に伸びる二本の突起にについて比較する．ほぼ同じ太さの突起であるが，分割法により描画の滑らかさが異なる．これは，分割された三角錐の細かさに起因する．①の分割法では各立方体が 6 個の三角錐に細分されるのに対して，②の分割法では 24 個の三角錐に細分される．また，よく見ると①の分割法において，格子の対角線（たとえば，図 4.11 における点 v0 と点 v2 を結ぶ線分）に対して，平行方向の場合は滑らかに描画されるが，垂直方向の場合は不自然な凹凸が生じている．

次に，分割数を同じにするのではなく，境界抽出を行う三角錐の個数が等しくなるように分割数を変えて比較してみる．①と②の分割法では，境界抽出を行う立方体あたりの三角錐の数は 1：4 である．よって，分割数の比は $\sqrt[3]{4}$ ：1 8：5 になっていればよい．図 4.12 と同じ条件で，分割数を 40 分割と 25 分割で比較した．

図 4.13 からも分かるように境界抽出を行う三角錐の数を等しくしたときも，①の分割法の場合，図 4.12(a)同様，方向による非対称性が生じている．そのため，DeltaViewer では，方法②を用いた．



(a)　　(b)

図 4.13

4.2. 三次元構築時における明暗値

分割方法，サンプルリスト，明暗値といくつもの工夫をして境界抽出した三次元データを考察すると，境界面に図 4.22(a)のような幾何学的な模様が見受けられた．この模様は，元の画像データには存在せず，上で示した分割法に由来する．この現象については，次のような変型した二分法を用いて解消した．

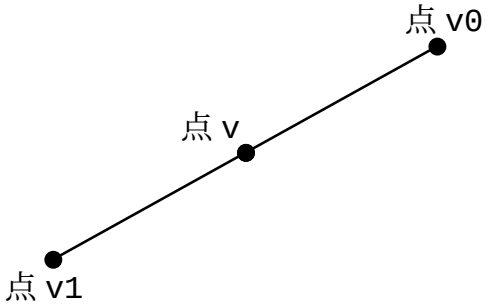


図 4.21

図 4.2 のように二点を v0，v1 とし，それぞれの点における明暗値を t0，t1 とする．また，二点を結ぶ線分と境界線の交点を v，点 v での明暗値を t とする．

点 v における明暗値 t の理想値は 0 である．t0 と t1 の絶対値の差が大きくなるにつれて，点 v における明暗値 t に誤差が生じる．そのため，｜t｜≦0.01 を満たすときの t を点 v における明暗値とみなす．以下にその計算方法を示す．

説明上，t0 は負，t1 は正であるとする．

① 点v0 と点 v1 を結ぶ線分と境界線との交点 v を求める

② 明暗値を求めるアルゴリズムを用いて，点v における明暗値を求める

③ t の値により誤差を調べる

⑴t > 0.01

```
v1 = v;          t1 = t;
```

⑵t < -0.01

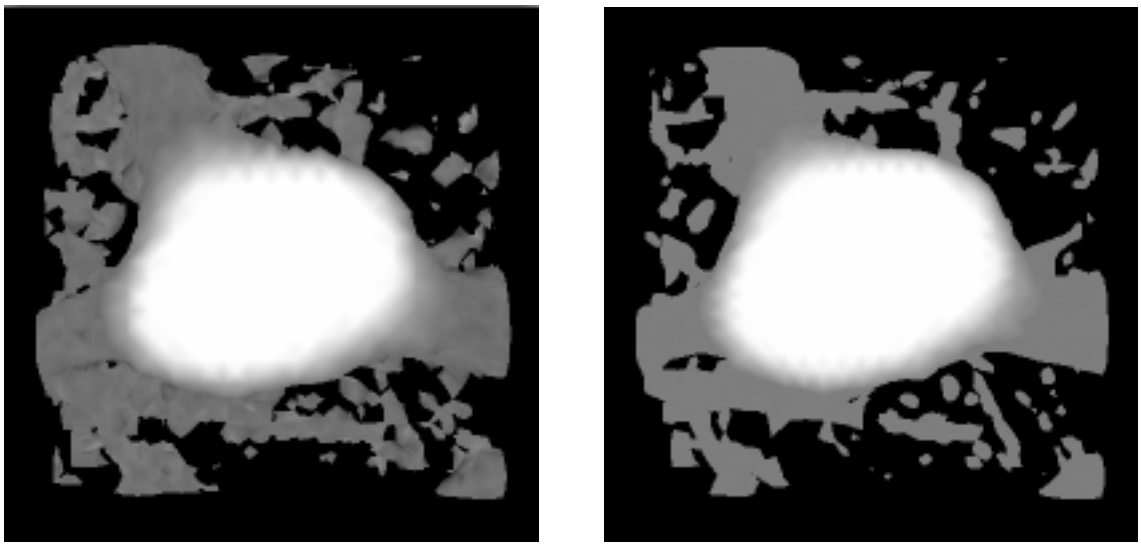
```
v0 = v;          t0 = t;
```

⑶-0.01 ≦ t ≦ 0.01

点 v を交点とみなす．

③において，-0.01 ≦ t ≦ 0.01 を満たさない場合は，新しく決まった点 v0 と点 v1 において，-0.01 ≦ t ≦ 0.01 を満たすまで上記の計算を再帰的にくり返す．

図 4.22 は，上で説明した方法を用いなかった場合(a)と用いた場合(b)の結果である．尚，一枚が 512*512（ピクセル）の画像データファイルを 60 枚読み込み，三角形四分割を用いて 25 分割したときの三次元構築したデータである．



(a)　　(b)

図 4.22

4.3．三次元構築にかかる時間短縮

初期段階の DeltaViewer において，三次元構築されたデータを得る時間が長かった．このデータを得る時間の短縮を考えた．

まず，分割された三角錐における境界面抽出までの計算過程をあげる．

① 各頂点の明暗値を求める．

② 明暗値をもとに境界面と交わるかを調べる．

③ ②において境界面と交わる場合，交点の明暗値 t が-0.01 ≦ t ≦ 0.01 を満たす交点を求める．

①～③において，必ず明暗値を求める計算を行う．また，分割された立方体もしくは三角錐において共有している頂点における明暗値は，重複して計算を行っている．

そこで，三次元構築されたデータを得る時間の短縮のために，次の三点を考えた．

- 読み込んだ画像データファイルから境界抽出を行うとき，必要最小の色データを順次メモリ上に確保する．
- 各頂点における明暗値を求める計算は一度だけとする．
- 境界面との交点を求める計算も一交点につき一度だけとする．

これら三点を実現するために，次のような操作を行った．

① 画像データファイルを読み込むことで得た三次元データをまず深さ方向に分割する．

② ①で分割された二次元平面の k 番目の二次元データをメモリ上に確保する．このデータを a とする．

③ k+1 番目の二次元データをメモリ上に確保する．この k+1 番目のデータを b とする．

④ ②で記憶した二面間において，幅と高さ方向でそれぞれ分割し，さらに分割された各立方体を三角錐に細分する．

⑤ 分割された三角錐において，境界面と交わるかを調べる．このとき，各頂点の明暗値と境界面との交点を求める計算は重複しないものとする．

⑥ k 番目の二次元データを消去し，k+1 番目の二次元データを a にコピーする．

⑦ k を１ずつ増やしながら，③～⑥の操作をくり返す．

以下に④と⑤で行っている重複計算の回避について詳しく述べる．

k 番目の二次元データ a を格子状に分割し，格子の頂点を（i，j）座標に割り当てる．尚，各格子の幅を di，高さを dj とする．（図 4.3 参照）

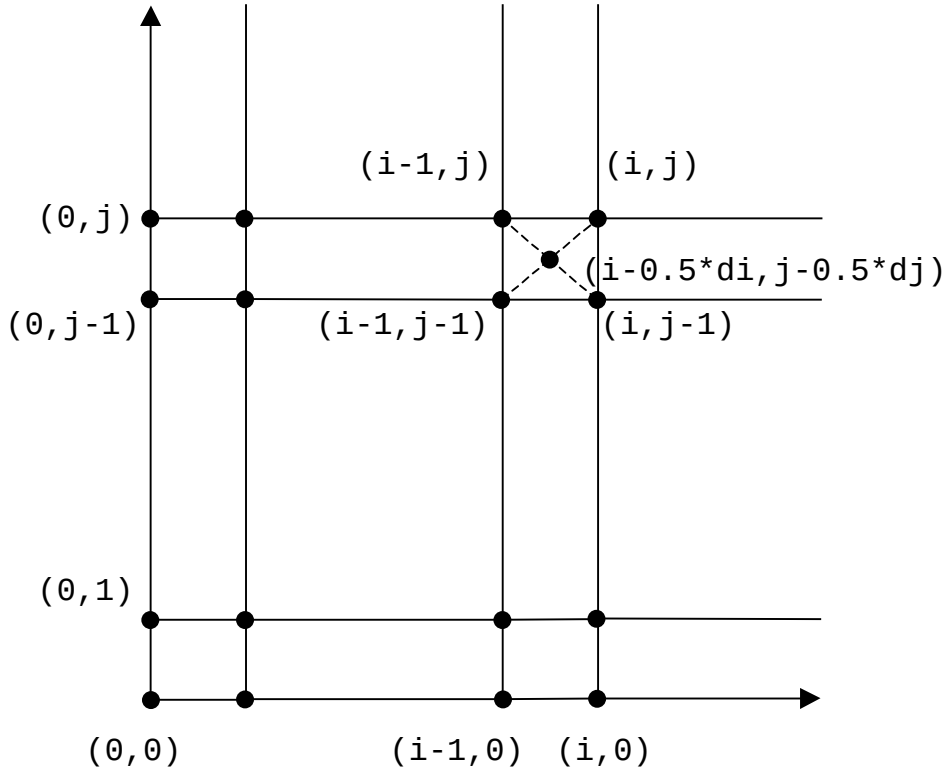


図 4.3

① i = 0，j = 0 のとき，つまり点（0，0）の明暗値を求める．

② i を 1 増やす．

③　点（i，0）の明暗値を求める．

④　点（i-1，0）と点（i，0）を結ぶ線分と境界線との交点を求める．

ただし，交点の明暗値 t は，-0.01 ≦ t ≦ 0.01 を満たすものとする．

⑤ i を 1 ずつ増やしながら，③と④をくり返す．

⑥ j を 1 増やす．

⑦　点（0，j）の明暗値を求める．

⑧　点（0，j-1）と点（0，j）を結ぶ線分と境界線との交点を求める．

ただし，交点の明暗値 t は，-0.01 ≦ t ≦ 0.01 を満たすものとする．

⑨ i を 1 増やす．

⑩　点（i，j）と格子の中心点（i-0.5*di，j-0.5*dj）の明暗値を求める．

⑪　点（i-1，j）と点（i，j）を結ぶ線分と境界線の交点を求める．
　点（i，j-1）と点（i，j）を結ぶ線分と境界線の交点を求める．
　点（i-1，j-1）と点（i-0.5*di，j-0.5*dj）を結ぶ線分と境界線の交点を求める．
　点（i，j-1）と点（i-0.5*di，j-0.5*dj）を結ぶ線分と境界線の交点を求める．
　点（i-1，j）と点（i-0.5*di，j-0.5*dj）を結ぶ線分と境界線の交点を求める．
　点（i，j）と点（i-0.5*di，j-0.5*dj）を結ぶ線分と境界線の交点を求める．
　ただし，それぞれの交点の明暗値 t は，-0.01 ≦ t ≦ 0.01 を満たすものとする．
⑫ i を 1 ずつ増やしながら，⑩と⑪をくり返す．
⑬ j を 1 ずつ増やしながら，⑦～⑫をくり返す．

K+1 番目の二次元データ b を上と同様に境界線との交点を調べる．このとき，上の操作に深さ方向を加えて，各立方体の共有点と境界面との交点における明暗値を重複計算しないようにする．境界面との交点を調べる計算も二度以上行わないようにする．

三次元構築されたデータを得る時間の比較表を表 2 に示す．ただし，一枚が 512*512（ピクセル）の画像データファイルを 121 枚読み込み，5 分割したときの結果である．尚，数値は 10 回行った平均値である．

表 2：重複計算回避による時間の比較(1/60 秒 / 回)

	alpha = 1.0	alpha = 3.0	alpha = 5.0
旧	211.7	2432.3	9572.8
新	82.0	462.3	1651.2

表 2 からも分かるように約 60%～80%の時間短縮を計れた．しかし，重複計算を回避するために三次元データを分割してから境界抽出までのプログラム行数が 472 行から 3778 行へと大幅に増えてしまった．

5．立体データの回転

三次元構築された立体データが大きくなるにつれて，マウストラッキングによる立体データの回転が滑らかでなくなる.そこで,OpenGL の機能である Cull Face と Display List を用いた.

Cull Face とは，クリッピングや隠面消去の効率をあげるための方法の一つである．例えば，立方体のように閉じた立体の場合，裏側にある面，すなわち視点に対して裏を向いている面は見ることができない．そういう面をあらかじめ取り除いておくことで，隠面消去の効率を上げることができる.

Display List とは，回転などの命令をすぐに実行せず，後で実行するために OpenGL コマンドを保存し，パフォーマンスを改善するものである.

Display List による回転の滑らかさを一命令にかかる時間で比較・検討した．ただし，一命令で３０度回転とし，２０回行った平均値をとっている.

表 3：Display List の有無によるデータ比較(1/60 秒 / 回)

	立体データ：1.7MB	立体データ：13.1MB
Display List：未使用	4.75	34.20
Display List： 使用	3.40	24.25

表 3 から分かるように，Display List を用いることで，約３０%の処理効率を上げることができた.

6. 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]OpenGL Architecture Review Board 編：OpenGL リファレンス，ピアソン・エデュケーション（1999）

[3]三浦憲二郎：OpenGL3D グラフィックス入門，朝倉書店（2000）

[4]M, Wada：A practical interpolation algorithm for small sample data，プレプリント

[5]T, Hogan：The programmer’s APPLE MAC sourcebook，Microsoft Press（1989）

参考 URL:

[URL1] Leica MICROSYSTEMS
http://www.leica-microsystems.com/

[URL2] 株式会社菱化システム
http://www.rsi.co.jp/

[URL3] Bio-Rad Laboratories Inc.
http://www.bio-rad.com/

[URL4] オリンパス光学工業株式会社
http://www.olympus.co.jp/

[URL5] カール　ツァイス株式会社
http://www.zeiss.co.jp/

[URL6] 株式会社ニコン
http://www.nikon.co.jp/

[URL7] TIFF ファイル形式
http://www.adobe.com/
http://www.libtiff.org/document.html

付録A

TIFF ファイルの構造

TIFF ファイルは８バイトのヘッダと IFD（Image File Directory）および色データで構成されている．

ヘッダには，IFD の先頭ポインタが書かれている．ヘッダ部分を表アに表す．

表ア：TIFF ヘッダの構造

Part	Size	Field
Header	2 bytes	Byte Order
	2 bytes	Version
	4 bytes	Pointer to first Image File Directory

IFD には，Directory Entry の数が記載され，その後，Directory Entry が個数分順に並んでいる．表イに IFD の構造を示す．

表イ：IFD の構造

Part	Size	Field
IFD	2 bytes	Number of directory entries
	12 bytes	First directory entries
	12 bytes each	Additional directory entries
	4 bytes	Point to next directory

各 Directory Entry は１２バイトで構成されており，それぞれの Directory Entry の始めの２バイトに Tag が記されている．Tag は，画像の幅，高さ，カラー形式などの画像の情報を表わしている．Tag が示す情報の値は，その Directory Entry 内の終わり４バイトで表示されている．もし，その値を４バイトで表示できない場合は，Directory Entry 外で記載するためのポインタを表記している．Directory Entry の構造を表ウに表わす．

表ウ：Directory Entry の構造

Directory Entry	Size	Item
	2 bytes	Tag
	2 bytes	Type
	4 bytes	Length
	4 bytes	Pointer to value

RGB Full-color 形式と Palette-color 形式における色データの構造について説明する．色データとは，画像のピクセルごとの色を表示している部分を指す．

各ピクセルでの色をピクセル値と呼ぶことにする．カラー画像においてピクセル値は，赤，緑，青の三色濃淡値で表わされている．濃淡値とは，２５６階調（もしくは１６階調）で色の割合を表わした数値である．三色ともに割合である数値が高くなると黒色に近付き，逆に数値が低くなりゼロに近付くと白色になる．

RGB Full-color 形式の色データは，ピクセルごとに赤，緑，青の三色の濃淡値が書かれている．またピクセル値は，赤，緑，青の各コンポーネントが R0,G0,B0,R1,G1,B1, R2,G2,B2,…,Rn,Gn,Bn の順に並んでいる．

Palette-color 形式の色データは，RGB Full-color 形式と異なり，ピクセル値ではなく，カラーマップへのインデックスが書かれている．カラーマップとは，ファイルごとに決められた赤，緑，青の三色濃淡値の表である．この三色は，R0,R1,R2,…,Rn,G0,G1,G2,…,Gn,B0,B1,B2,…,Bn のようにインデックス（0,1,2,…,n）の順に並んでいる．尚，インデックスが i の場合は，カラーマップの i 番目の赤，緑，青の濃淡値がそのピクセル値となる．

TIFF ファイルには，DeltaViewer で対応している RGB Full-color 形式と Palette-color 形式以外に Bilevel 形式と Grayscale 形式がある．Bilevel 形式は白色か黒色の二色で表され，Grayscale 形式は白色から黒色の濃淡値で表されるものである．

付録Ｂ

DeltaViewer (version1.3.1)　使用説明書

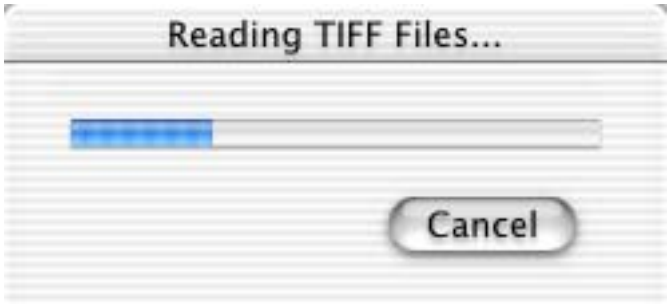
DeltaViewer は，連続な２次元断面画像から任意の境界を抽出して曲面データを３次元構築し，OpenGL を用いてコンピュータ画面上に表示するソフトウェアです．使用手順は，大きく分けて以下の４ステップになります．

STEP　１：画像ファイルの読み込み

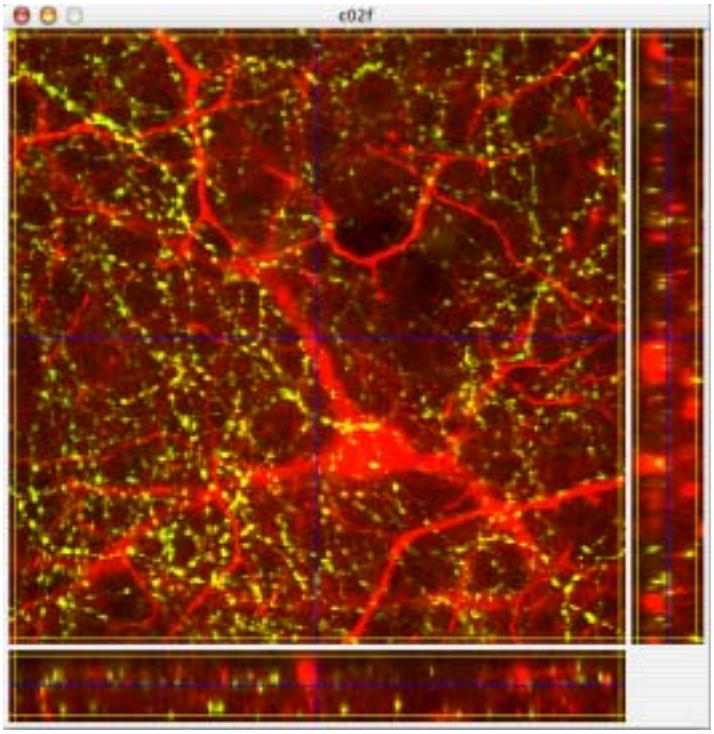
DeltaViewer で読み込める画像ファイルは，現在のところ TIFF 形式のみです．カラー形式は，RGB の各色の成分ごとに別ファイルになっているものと，RGB 三色が同一ファイルになっているものに対応しています．また，ファイル名は，“ abcd001.TIF ”などのように先頭部分（abcd），通し番号（001），拡張子（.TIF）の三つのパートからなるものと仮定しています．一枚目の画像を読み込むと先頭部分，通し番号の桁数，拡張子は決定し，abcd001.TIF, abcd002.TIF,abcd003.TIF,… のように昇順で通し番号の最後まで読み込んでいきます．先頭部分の長さや通し番号の桁数に特別な制限はありません．

実際の操作としては，SectionView コマンドから OpenTIFF(color)を選び, 読み込む先頭ファイルを指定するだけです. この color には, Red, Green, Blue, RGB の４種類あります．初めの３つは RGB の色ごとにファイルが分かれている場合に対応するもので，最後の１つは RGB 三色が同一ファイルになっている場合に用いるものです．一番目に赤色の画像データを読み込むときは OpenTIFF(Red) を選び，続けて緑色の画像データを読み込む時は OpenTIFF(Green)を，青色の画像データを読み込む時は OpenTIFF(Blue) を選択し，読み込む先頭ファイルを指定します．同じ色は二度以上選べません．もし OpenTIFF(RGB)を一番始めに選択すると RGB 三色とも選んだ状態になるので，他色は選択できません．選べない色については，グレーアウトされるので，間違って同じ色を選択することはないはずです．

読み込み始めると下図のようなプログレスバーが表示されます．



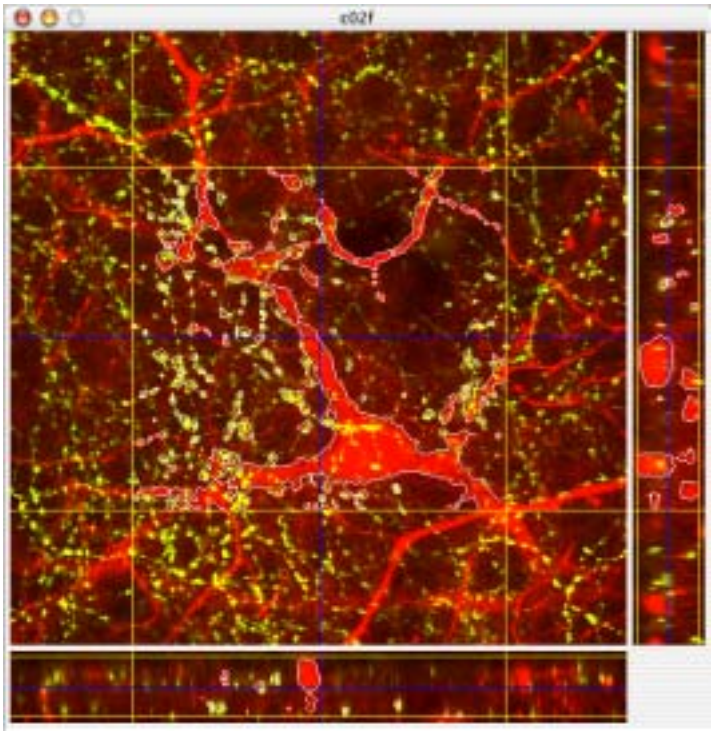
読み込みが終わると,ウィンドウが表示されます. このウィンドウを SectionView ウィンドウと呼びます.



SectionView ウィンドウには，２次元の画像ファイルを複数読み込んで得られる３次元のデータを正面，右側および下側から見たときの断面図が描画されます．最初は，正面から見た断面図中央の縦の青線に沿って切ったときの断面図が右側に描画されています．その他の断面図も同様に青線での断面図が描画されます．違う場所の断面図を見たいときは，SectionView ウィンドウ内にある青線を動かします. マウスポインタは青線の近くにいくとに変わり，クリックするとに変わりますので，そのままマウストラッキングすると，青線の移動に伴い，移動先の断面図が逐次描画されます.

STEP　2：２次元画像上に境界線を書く

境界抽出した時に内側になる色と外側になる色をいくつかマウスクリックで指定するだけで境界線が描けます．内側の色を指定するときは，Option を押すとポインタが  に変わるので，そのままマウスクリック，外側の色を指定するときは，Option と Shift を同時に押すとポインタが  に変わるので，そのままマウスクリック．それぞれ内側と外側になる色をひとつ以上マウスクリックすると SectionView ウィンドウ内に白線で境界線が書かれます．内側の色と外側の色を指定するたびに境界線が変わり，その都度描画されます．



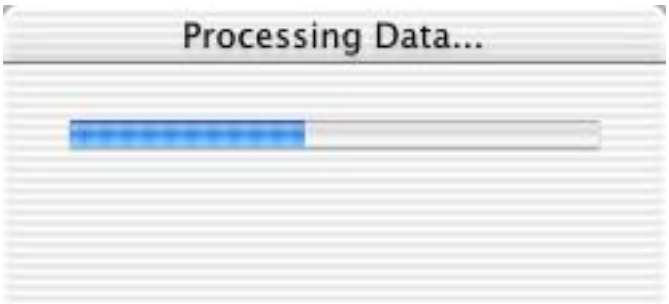
境界線に影響するコマンドとしては，Smoothness，Division および Orientation があります．Smoothness のデフォルトは alpha = 1，Division のデフォルトは 10 です．alpha の値が大きくなると三次元構築したときの物体の色合いがよりスムーズに描画されます．しかし，alpha の値が大きくなると計算時間が長くなるため，境界線を引くまで，もしくは三次元構築を終えるまでの時間が長くなります．Division の値を大きくするとより詳細な境界線を書くことができます．しかし，三次元構築された時のデータが大きくなり，三次元構築された立体データを滑らかに動かすことができなくなります．Orientation メニューは，SectionView ウィンドウ内に描画されて

いる断面の向きを変えます．Flip Left/Right で左右が, Flip Top/Bottom で上下が，Flip Front/Rear で奥行き方向が入れ替わります.

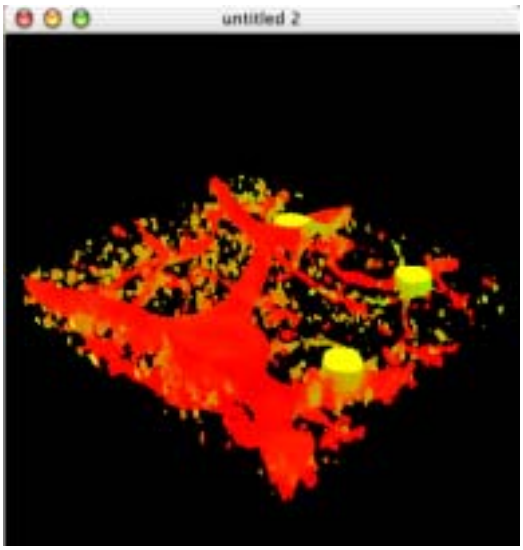
STEP　3：境界を抽出し３次元構築する

SectionView ウィンドウ内にある黄線で囲まれている範囲で境界抽出をするので，部分的に三次元構築するときは，STEP1 で青線を動かした時と同じようにマウストラッキングで黄線を動かします．ウィンドウ全体のデータを取るときも同じように黄線を動かします．境界抽出する範囲が決まれば，SectionView コマンドの 3DView Window を選びます.

３次元構築が始まると下図のようなプログレスバーが表示されます.



３次元構築が終わると新しいウィンドウにそのデータが表示されます．このウィンドウを 3DView ウィンドウと呼びます.

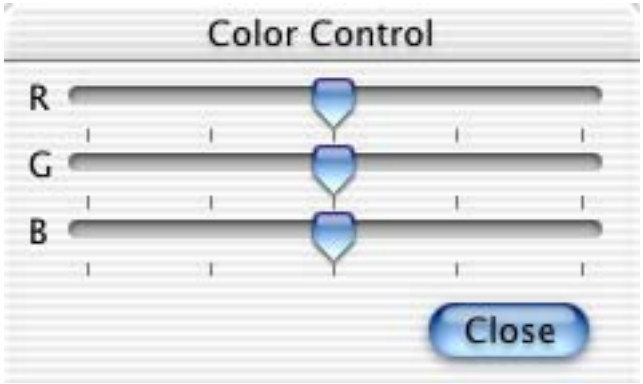


STEP　4：３次元構築した立体データを回転する

3DView ウィンドウでのポインタは，通常になっており，クリックすると に変わります．のままマウストラッキングすると立体データが回転します．トラッキング中にマウスから指を離すと立体データは回転し続けます．回転をとめるには，3DView ウィンドウ内でマウスクリックします．

3DView ウィンドウ内に表示されている立体データを保存するときは，3DView コマンドの SaveAs を選択し，ファイル名を決めます．一度，保存したデータを 3DView ウィンドウに表示するときは，3DView コマンドの Open を選び，ファイルを指定します．

3DView のコマンドとして，立体データの色のトーンを変えることができます．3DView コマンドの Color Control を選ぶと下図のようなウィンドウが表示されます．



RGB それぞれのつまみをマウストラッキングで移動させると，トーンを変えることができます．左側に動かすと暗くなり，右側に動かすと明るくなります．立体データの色合いを見て，つまみを調節します．Color Control ウィンドウを閉じるまで他の操作はできないので，調節を終えると，Color Control ウィンドウを閉じて下さい．

付録C

DeltaViewer (version1.3.1)　Menu List

SectionView :

Open TIFF(color)

・画像データファイルを読み込む

・color の種類は，Red,Green,Blue,RGB の４種類

・SectionView ウィンドウで同じ色を二度以上選べない

（一番目に Red を選び，続けて異なる色を読み込む場合は，Green,Blue のみ選択可能である．もし, RGB を一番目に選ぶと他色は選択できない.）

Smoothness

・ 境界線を引くとき，及び３次元構築するときにピクセル値を求めるために用いる補間係数

（alpha の値が大きくなるとよりスムーズに描かれるが，補間に要する時間が長くなる．詳しくは，3.6.分割法とピクセル値を参照.）

・デフォルトは，alpha = 1

Division

・　画像の分割数

（分割数が大きくなると３次元構築されたときのデータが大きくなるため，立体データの回転が滑らかでなくなる.）

・　デフォルトは,10

Orientation

・ SectionView ウィンドウ内に表示されている断面画像の向きを変える

・ Flip Left/Right：左右を逆転する

・ Flip Top/Bottom：上下を逆転する

・ Flip Front/Rear：前後を逆転する

Open 3DView Window

・ SectionView ウィンドウ内の黄線で囲まれた部分のみを抽出して，三次元構築を行う

・　３次元構築後，新しいウィンドウに立体データを表示する

3DView :

Open

・　ファイルに保存されている３次元構築データを読み込む

SaveAs

・3DView ウィンドウに表示されている３次元構築されたデータをファイル保存する

Color Control

・3DView ウィンドウに表示されている３次元構築されたデータの色の明暗を RGB ごとに調節する