

# Volume Catcher: ポリュームセグメンテーションのためのユーザーインターフェース An User Interface for Volume Segmentation

大和田 茂\*  
Sony CSL

Frank Nielsen†  
Sony CSL

五十嵐 健夫‡  
東京大学 / JST PRESTO

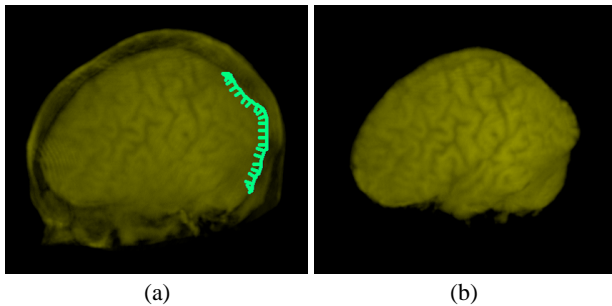


Figure 1:  $105 \times 73 \times 78$  の頭部 MRI データに本システムを適用した例. ポリュームレンダリング画像上において、脳の輪郭の一部を二次元的になぞることにより (a), 脳全体領域を取り出すことができる (b). 奥行き情報はシステムが自動的に計算する.

## Abstract

本稿では *Volume Catcher* と呼ばれる、ポリュームレンダリング画像に対して二次元的にユーザーが関心のある領域 (region of interest, ROI) の輪郭線をなぞるだけでセグメンテーションを行うユーザーインターフェースを提案する. ユーザーが描いたストロークは、視点情報およびデータの分析によって自動的に三次元化されるため、対象を切断するなどして明示的に奥行きを指定する必要がない. 三次元化されたストロークの周囲には前景および背景領域を示す束縛点集合を発生させ、セグメンテーション自体は既存のアルゴリズムによって行う [Nock and Nielsen 2004].

## 1 背景

ポリュームセグメンテーションは、ポリュームデータを視覚的あるいは構造的に意味のある領域に分割する画像処理である. これは生データから有用な情報 (形状や体積など) を得るために不可欠なプロセスであり、それゆえ長きにわたって研究されているが、全自動の手法は今もって存在しない. なぜなら、人間が期待するセグメンテーション結果を得るには、エッジ検出やテクスチャ分析などの低次の情報から、物体の全体形状やトポロジーなどの高次の情報まで、幅広く高度な情報処理が必要だからである. 特に高次の情報を扱うことは難しい. 例えば、テーブルの上に皿と、その上に葡萄が乗っている画像が与えられたとする. ここで、ユーザーが欲するセグメンテーション結果は、葡萄の粒一つ一つが分離した状態かもしれないし、葡萄全体を一つの領域として得たいのかもしれない. あるいは、葡萄と皿全部をまとめて一つの領域として出力することが正解であることもあり得る. この種の高次の情報は、ユーザーの指示なしに自動的に生成することは不可能であると考えられる.

ユーザーの見地からすれば、このような情報をどう与えるかが問題となる. マウスなど通常の入力デバイスは二次元情報しか与えられないので、三次元のポリュームデータを対象とする場合に、不足する一次元をどう与えるかという問題があるから

である. これまでの手法では、このような情報は対象を切断した切断面上に与えていた [Tzeng et al. 2003; Sherbondy et al. 2003]. 我々はよりシンプルに、ポリュームレンダリングされた画像に直接ストロークを描くことにより ROI を指定することを提案する (Figure 1). ユーザーが自分が関心のある領域の輪郭を二次元のストロークでなぞると、システムは、そのパスが三次元的にはなるべくポリュームのシルエット領域、すなわち、グラディエントが視線方向に垂直に近い領域を通るように奥行きを付加する. この三次元化されたパスの周囲に前景および背景の束縛点が自動的に発生され、既存のポリュームセグメンテーションアルゴリズム [Nock and Nielsen 2004] の入力とされる.

## 2 アルゴリズム

### 2.1 二次元のストロークから三次元のパスへ

我々のアルゴリズムは、まず最初にスクリーンにユーザーが描いた二次元のストロークに奥行きを付加して三次元にする. ここで、ROI の輪郭は視覚的にきわだっており、ユーザーはその輪郭の近くをなぞったと仮定する. この仮定が成り立つならば、三次元のパスは、ROI のシルエットの近くを通っているはずである. 言い換えれば、ポリュームデータのグラディエントが、視線方向に対しほぼ垂直になっている. このようなパスを、我々は次のようにして求める.

1. 二次元のパスを奥行き方向にスイープし、三次元の曲面を作る (Figure 2a). この曲面をスイープ曲面と呼ぶ. 我々の実装では、この曲面は四角形ポリゴン集合として表現される.
2. スイープ曲面に座標系を設定する. システムは、奥行き方向が  $x$ 、スクリーンに平行な向きが  $y$  となるように座標系を設定する (Figure 2b). すなわち、視点に近い方のスイープ曲面の端は  $x = 0$  であり、二次元パスの始点に対応する奥行き方向の直線は  $y = 0$  となる.
3. スイープ曲面上にサンプリング点の集合を設定する. サンプリング点はパラメータ空間における格子点である. 我々の実装では、この格子間隔はポリュームデータのバウンディング球の直径の 0.3% の幅に設定されている<sup>1</sup>. 以下の文脈では、各格子点を  $L_{ij}$  と表現する. ( $i, j$  は  $x, y$  方向のインデックスであり、 $0 \leq i \leq X_{max}, 0 \leq j \leq Y_{max}$  である)
4. 各格子点  $L_{ij}$  において、シルエット係数  $S_{ij} = |N_{ij} \cdot G_{ij}|$  を計算する. ここで、 $N_{ij}$  は正規化されたスイープ曲面の法線であり、 $G_{ij}$  は正規化されたポリュームのグラディエントである (Figure 2c). シルエット係数  $S_{ij}$  は、その点が現在の視点からどの程度シルエットに見えるかという度合いを示す.

ポリュームのグラディエント  $G_{ij}$  を求めるには以下のようにする. まず、元データが色情報を持っている場合には、 $Gray = Alpha \times (0.299 \times Red + 0.587 \times Green + 0.114 \times Blue)$  という式を使ってグレイスケールに変換する [Russ 2002]. 伝達関数などで、元のポリューム値がフィルタリングされている場合には、フィルタ適用後の色を用いる. さらに、ノイズ

\*Shigeru Owada: sowd@acm.org

†Frank Nielsen: frank.nielsen@acm.org

‡Takeo Igarashi: takeo@acm.org

<sup>1</sup>厳密に言えば、透視投影の場合にはこのパラメータ化は、三次元空間内では一様とはならないが、我々はこの影響を無視し、パラメータ空間内で一定間隔の格子を用いることとした.

の影響を低減するために、スムージングを行う。スムージングにはガウス関数を用いた。関数の半径  $R$  は、ユーザーの描くストロークのエラー許容量と一致させる (我々の場合はスクリーン上の 5 ピクセル分としている)。グラディエント計算とスムージングを同時に行うために、ガウスカネルを各軸方向に前進差分したものをを用いて畳み込み計算を行うこととした。

このように、ぼかしカーネルのサイズをレンダリング画像領域で定めることにより、ユーザーは対象を大きく拡大した時には細かい操作をすることができ、逆に拡大率を下げた時には大域的な構造を対象にすることができることになる。ただし、拡大率を上げすぎると、カーネルのサイズが小さくなりすぎ、ノイズに弱くなるという欠点もある。

- ユーザーが描いたストロークの奥行きを計算する問題は、前項で計算した格子状において、辺  $y = 0$  上の点からはじまり  $y = Y_{max}$  に至るパスのうち、パス上のシルエット係数の総和が最大になるものを求める問題に帰着できる (Figure 2d)。このパスは、各  $y$  に対して  $x$  の値が一つしかないので、関数  $x = f(y)$  のように表現できる。パスに連続性を持たせるため、我々は  $|f(y_i) - f(y_{i+1})| \leq c$ 、という条件を課すことにした (我々の実装では  $c = 1$  である)。この問題を解くには動的計画法を用いた [Cormen et al. 2001]。すなわち、動的計画法計算用の格子を用意し、格子点間  $\{L_{pq} : |p - r| \leq c, |q - s| = 1\}$  を枝で連結し、 $y = 0$  上の一列の格子点を対応するシルエット係数で初期化し、あとは Dijkstra 法を用いて最適パスを求める。最短経路を求める問題と異なる点は、我々の手法ではコストが最小ではなく最大になるパスを求めているという点と、枝ではなく頂点にコストが設定されているという点である。さらに、閉じたストロークに対してはもう一つ、 $|f(y_0) - f(y_{max})| \leq c$  という条件も与えた。開いたストロークに対しては、動的計画法は一度だけ行えばよいが、閉じたストロークに対しては、条件  $|f(y_0) - f(y_{max})| \leq c$  を満たすため、辺  $x$  方向のサンプル点の数だけ計算を行わなければならない ( $y = 0$  上の各点を起点とする Dijkstra 法を適用するのに近い)。

パラメータ空間で見つかった最適パスは、対応する三次元位置を与えられて三次元のパスとなる (Figure 2e)。Figure 1 の時に計算に使われたシルエット係数の格子  $S_{ij}$  と、見つかった最適パスが Figure 3 である。

## 2.2 束縛点の生成とセグメンテーション

三次元のパスが求まったら、次にセグメンテーションに必要な束縛点集合を発生させる。点集合は、パスをオフセットした位置に発生させる。オフセット方向  $D$  は、視線ベクトルとパスの接線ベクトルの外積の方向である (Figure 4a)。オフセットされる距離  $e$  は、前項で用いたスムージングカーネルの半径  $R$  に比例させる。従って、ストロークのエラー許容量にも比例することになる (我々の実装では  $e = 2R$  とした)。まとめると、三次元パスは、 $\pm e \frac{D}{|D|}$  だけオフセットされ、ストロークの右側には前景の束縛点が、左側には背景の束縛点が生成される (Figure 4b,c)。Figure 1 に対して生成された束縛点を Figure 5 に示した。セグメンテーションには、我々は Nock らによるアルゴリズムを用いた [Nock and Nielsen 2004]。この手法では、各ボクセル (二次元の場合はピクセル) が別の領域に含まれる状態から始まり、統計的基準により隣接する領域を繰り返し併合していくことでセグメンテーションを行うものであり、この併合をコントロールするために前景と背景の点集合を入力することができる (統計量を操作するために、オリジナルの論文では constraint ではなく bias という呼称が使われている)。これ以外にも前景と背景の拘束条件をユーザーが与えることによってセグメンテーションを行うアルゴリズムは多い [Li et al. 2004]。

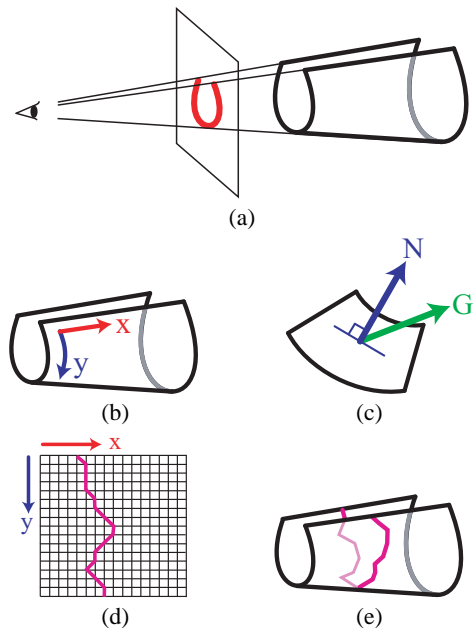


Figure 2: 二次元の入力ストロークから三次元のパスへのアルゴリズム。 (a) スweep曲面の構築。 (b) スweep曲面上でのパラメータ設定。 (c) シルエット係数の計算。 (d) 最適パスの発見。 (e) 得られた三次元のパス。

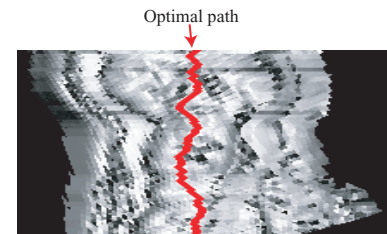


Figure 3: Figure 1 で使われたシルエット係数の格子と、計算された最適パス

実際のセグメンテーションアルゴリズムにはパスの情報ではなく、この束縛点を与えられる。もしユーザーが与えたストロークが閉じていたならば、スweep曲面の内部領域のみが結果として出力される。

## 3 結果

Figure 6 は、high-potential iron protein と呼ばれるデータに対して本システムを適用した例である。我々のシステムは、ROI が一部しか見えていない時にも正しく動作する (Figure 6b)。また、カラーデータに適用した例が Figure 7) である。一般に、色情報を持つボリュームデータを得ることは難しい。我々は小川らのスライス装置 [Ogawa et al. 1999] を用いることにより、直接物体をスライスしてカラーのボリュームデータを得た (Figure 8)。

提案システムを用い、Intel Xeon(TM) 3.2-GHz CPU と 2GB の RAM を搭載したデスクトップマシンを用いて、三人の被験者に作業を行ってもらった。タスクの内容は、このインターフェースを用いて様々な任意の領域を 20 回選択するというものである。対象データは二種類 (high-potential iron protein (66 × 66 × 66) と頭部 MRI データ (105 × 73 × 78)) とした。このテストの結果を Figure 9 に示した。このグラフの縦軸はストロークの三次元化と束縛点発生にかかった時間の合計で、横軸はスweep曲面上

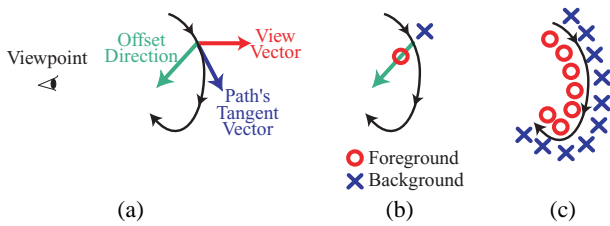


Figure 4: 束縛点の発生

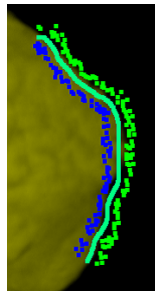


Figure 5: Figure 1 の場合に生成された束縛点集合

のサンプル点の数, すなわち, ユーザーが描いたストロークの長さに比例する量である. このグラフから, たいていの場合 2 秒以下で処理が完了することがわかる. また, 閉じたストロークの三次元化には動的計画問題を何度も解く必要があるにもかかわらず, ストロークが開いているか閉じているかは実行時間にほとんど影響を与えないということもわかった. これは, 実行時間のほとんどはグラディエントを計算するための畳み込み演算に費されるためであると考えられる. なお, セグメンテーションそのものにかかる時間は表には示されていない. なぜなら, この時間はどのアルゴリズムを用いるかに大きく左右されるからである. [Nock and Nielsen 2004] を用いた我々の場合は, 発生される束縛点の数と, ボリュームデータの量に応じて 0.5 秒から 10 秒程度の時間がかかった.

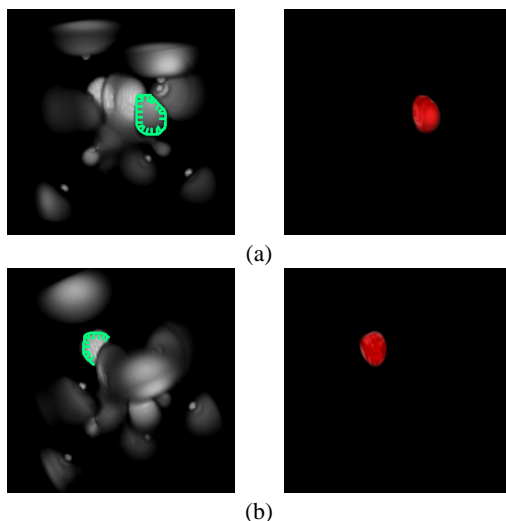


Figure 6: サイズ  $66^3$  の High-potential iron protein データに適用した例. セグメンテーションされた領域は不透明で赤いボクセルとしてボリュームレンダリングにより提示した.

また, この手法がうまく働かない例がいくつか存在する. まず, 現在の視点から見て, ほぼ同じシルエットを共有する領域が二個

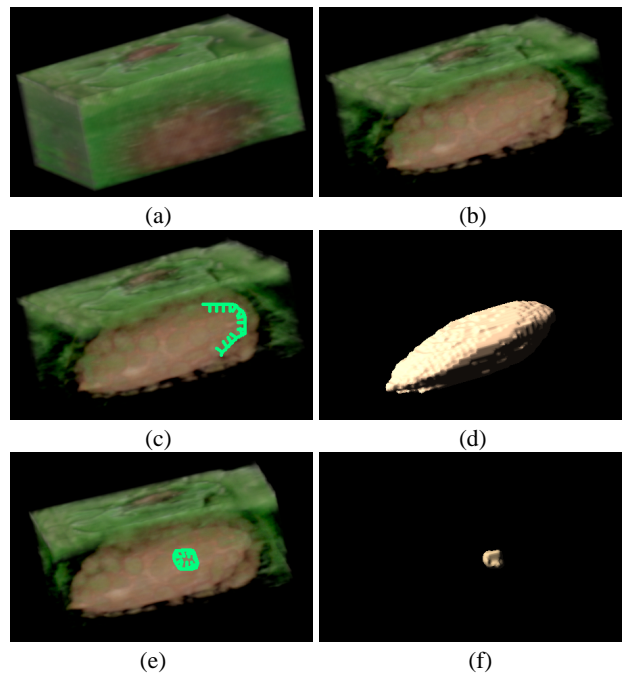


Figure 7: 色付きデータ ( $126 \times 89 \times 53$ , チョコレートクリスピのデータ) に本システムを適用した例. まず元データに伝達関数を適用して (a) アーモンドを見やすくする (b). ユーザーがストロークを描くと (c), アーモンド領域が取り出され, 境界面が表示される (d). この例では, ROI は Marching Cubes 法を用いてポリゴンリングを行った. 表面の色は, 取り出された領域の色の平均としている. より小さなクリスピ部分も取り出すことができる (e, f).

以上ある場合には, 片方しか選ばれない (Figure 10a). また, 対象領域が薄いなどの場合に, ROI 内に入るべき束縛点が ROI の外側に出てしまうことがある (Figure 10b). また, ROI と似た特徴を持った領域が隣接している場合, ROI 外になければならない束縛点が, 望まない方の領域に入ってしまうことがある (Figure 10c). この場合, 用いたセグメンテーションアルゴリズムがグローバルな位置情報を考慮しない場合には正常に働かなくなる恐れがある. しかし, これらの問題は, 視点を変えることにより解決できることがほとんどである.

ただし, 用いたセグメンテーションアルゴリズムの性能にはもちろん影響される. 従って, パスの三次元化が成功しても, セグメンテーションアルゴリズムの欠点により望む結果が得られないこともある.

#### 4 今後の展望

我々の実装は非常に基本的なものであり, 様々な拡張が考えられる. 例えば, このユーザーインターフェースを用いて, 伝達関数設計の助けとすることが考えられる. すなわち, ユーザーが強調したい部分を本手法を用いて直感的に指定し, 伝達関数のパラメータを操作するということである. これにより, 試行錯誤の手間が軽減すると期待される. 次に, Snakes など様々なセグメンテーションアルゴリズムを組み合わせることで精度の向上を図りたい. ここでの問題は, 場合によっては束縛点とは異なる情報を要求されることである. 例えば Snakes であれば, 初期バウンダリが必要となる. また, ユーザーが意図した結果が得られなかった時にそれを修正する操作も必要である. さらに, 我々の実装では Nock らのセグメンテーションアルゴリズムを用いたが [Nock and Nielsen 2004], この手法ではハードセグメンテーションしか行うことができない (各ボクセルが前景に属するか背景



Figure 8: A cutting device

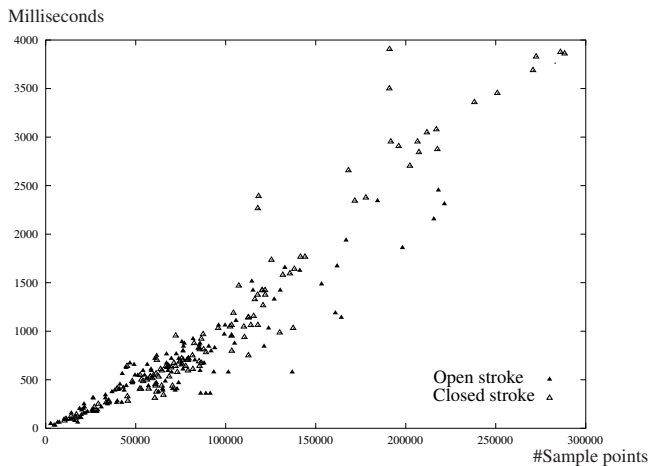


Figure 9: High-potential iron protein データおよび頭部 MRI データに対し、本システムを適用した結果のグラフ。この縦軸は、ユーザーが与えた二次元のパスを三次元化する時間、および、セグメンテーションアルゴリズムの入力となる束縛点集合を発生させるのにかかった時間である。セグメンテーションそのものにかかった時間は含まれていない。

に属するかの二値で得られる)。将来的には、Alpha Matting の手法 [Sun et al. 2004] を用いるなどして不明確な境界も扱えるようにしたい。

## References

- CORMEN, T. H., STEIN, C., RIVEST, R. L., AND LEISERSON, C. E. 2001. *Introduction to Algorithms*. McGraw-Hill Higher Education.
- LI, Y., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Lazy snapping. *ACM Trans. Graph.* 23, 3, 303–308.
- NOCK, R., AND NIELSEN, F. 2004. Grouping with bias revisited. In *IEEE International Conference on Computer Vision and Pattern Recognition*, IEEE CS Press, A. B. L.-S. Davis, R. Chellapa, Ed., 460–465.
- OGAWA, Y., OHTANI, T., SUGIYAMA, J., HAGIWARA, S., KOKUBO, M., KUDOH, K., AND HIGUCHI, T. 1999. Three dimensional visualization of internal constituents in a rice grain. *ASAE/CSAE-SCGR Annual International Meeting*, 993059.

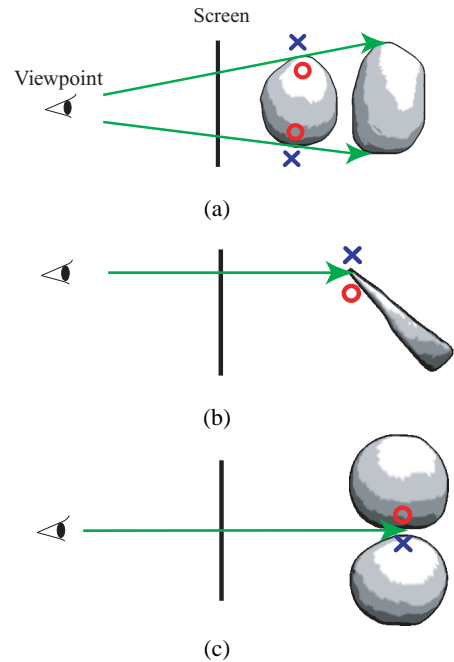


Figure 10: うまくいかない例。(a) 二つの領域が同じ輪郭線を共有する。(b) 対象領域が薄い。(c) 類似の構造を持った領域が隣接している (これらの例では、対象領域は境界ポリゴンで表現されている)。

RUSS, J. C. 2002. *The Image Processing Handbook Fourth Edition*. CRC Press.

SHERBONDY, A., HOUSTON, M., AND NAPEL, S. 2003. Fast volume segmentation with simultaneous visualization using programmable graphics hardware. In *Proceedings of IEEE Visualization 2003*, IEEE, 171–176.

SUN, J., JIA, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Poisson matting. *ACM Trans. Graph.* 23, 3, 315–321.

TZENG, F.-Y., LUM, E. B., AND MA, K.-L. 2003. A novel interface for higher-dimensional classification of volume data. In *Proceedings of IEEE Visualization 2003*, IEEE, 505–512.