グラフィクスとCAD 101－1
（2000. 12. 8）

ストロークによるペンアンドインク風画像の生成とそのアニメーション化

芳賀俊之　　西田友是

東京大学大学院 理学系研究科 情報科学専攻
E-mail : {haga, nis}@is.s.u-tokyo.ac.jp

ペンアンドインク風の画像は、写真のようにリアルな画像よりも、詳細を抽象化し、形状を強調させる等の特徴を持つために注目されている。そのような利点のため、ペンアンドインク風の画像は三次元モデルのアニメーションの描画にも適用できる。描かれるイラストのストローク (筆運び) に、元の三次元の幾何的情報が反映されるようにしつつ、ペンアンドインク風の線画を生成するシステムを提案する。また、対象となるモデルに回転もしくは動きがあるときには、生成される前後のフレーム間でストロークに相関関係を持たせ、自然なアニメーションが得られるようにしている。提案するシステムでは、以下のように画像の生成を行う。(i) 三次元幾何モデルから、濃淡を表すグレイスケール画像と、それぞれの面上での筆運びの方向の情報と全体の輪郭線の情報を取り出す。(ii) それらと以前のフレームの情報を基に、方向付けされた線を描くが、その際、線画の濃淡画像と、グレイスケール画像とを比較し、線を描く適切な位置を決定する。

Generation and Animation Methods for Pen-and-Ink Illustrations by Using Strokes

Toshiyuki Haga　　Tomoyuki Nishita

Department of Information Science, Graduate School of Science,
University of Tokyo
E-mail : {haga, nis}@is.s.u-tokyo.ac.jp

Pen-and-ink illustrations are attractive for having more abilities than photorealistic images to mute detail, clarify shapes and so on. Because of these advantages, pen-and-ink illustrations can be used to create 3D model animations. We have developed a system for creating pen-and-ink-style line drawings in which the strokes of the rendered illustration follow the features of the original 3D geometry. If the model is rotating/moving, the system makes the coherence between frames for getting smooth animation. Our system generates an image as follows. (i) Based on the 3D geometry model, we obtain a greyscale image that conveys the tone, the information of the direction of the stroke on each face of an object, and the information of the object's silhouettes. (ii) Using the information we got from (i) and that of previous frames, we compare the tone image of the line drawing image with the greyscale image and draw directed lines of the region where the tone is not sufficient.

1 はじめに

近年、ペンや鉛筆による線画や水彩、油彩の様式等を取り入れた画像の生成を行うノンフォトリアリスティックレンダリングの研究が盛んである。ペンアンドインク風画像の生成についての研究もそのひとつである。ここで、ペンアンドインク風画像とは、物体の濃淡をストローク (ペンで描かれた線) の密度で表し、表面の形状等をストロークの方向や描き方により表現するものである。そのような性質から、写真のようにリアルな画像よりも、詳細を抽象化し、形状を強調させる等の特徴も持っている。

ペンアンドインク風画像は、その入力データの違いにより、大きく二つに分類できる。一つは、二次元のグレイスケール画像を入力とする image-based システム、もう一つは、三次元幾何モデルを入力とする geometry-based システムである。image-based システム [1, 5] の主な利点としては、任意の画像を入力として扱えることがある。しかし、三次元情報を伝えるためには、ユーザーが見えない線を描いたり、画面上でストロークの集合の描画の方向を指定することが必要となる。一方、geometry-basad システム [2, 4] では、三次元の幾何学的、視覚的情報を利用することができるおかげで、そのシーンでの表面の色調やテクスチャを伝えるだけでなく、表面の自然な輪郭に沿って筆を運ぶことで、表面の三次元形状をも伝えるようなイラストを生成することが可能である。また、滑らかな表面形状を伝えるためのストロークの方向場の生成法も考案されている [4]。

提案法は、三次元幾何モデルを対象として扱う geometry-based システムである。また、従来あまり論じられることのなかった、ペンアンドインク風画像のアニメーションの生成についても考慮したものである。

2 描画アルゴリズム

本節では、一枚一枚の画像の生成プロセスについて述べていく。アニメーション化の手法については 3 節で説明する。画像生成の大まかな流れは以下の通りである。入力データとして三次元ポリゴンモデルを扱い、光源としては平行光源を用いている。また、生成画像は黒色のペンで描かれたものである。

1. Z バッファ法を用いて、物体の特徴的情報を保持する 3 つの画像の生成を行う。
2. それらの画像を基に、物体の濃淡や形状に従って、描画のためのストローク生成を繰り返す。
   i. 新たにストロークを置く位置、両端点を計算する。
   ii. ストロークにフィルタをかけて、ぼかした画像を生成する。
   iii. ストローク密度の判定を行い、描画するか否かを決定する。
3. 輪郭・稜線と生成されたストロークを実際のスクリーンへ描画する。

以下、それぞれの詳細を説明する。

2.1 特徴的情報を保持する画像の生成

対象であるモデルと視点、光源を定めた後、Z バッファ法を用いることで、ストロークの生成で使用するための、物体の特徴的な情報を保持する 3 つの画像、すなわち、(i) 物体表面の濃淡を表す **tone-image** (図 1 左上)、(ii) それぞれの場所でのストロークの方向を表す **direction-image** (同右上)、(iii) 輪郭や稜線の情報を保持する輪郭画像 **silhouette-image** (同左下) の生成を行う。なお、**direction-image** では方向 (スクリーン上での角度) を色として保存している。

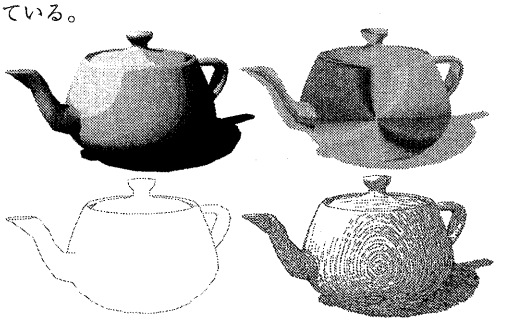

図 1: 3 つの画像と結果の生成画像 (右下)

tone-image は、面の法線ベクトルと光線の方向ベクトルから、ランバートの余弦則に従って生成され、**direction-image** の各画素に保持される方向の値は、面の法線ベクトルと視線ベクトルの外積ベクトルをスクリーンに投影したベクトルより計算される。**silhouette-image** では、視線に対して表向きの面と裏向きの面の間の辺を輪郭とし、ある辺と隣接する面の法線ベクトル同士の内積の値が閾値以下であるときに稜線として、描画する。また、**tone-image** には、物体が物体に落とす影の部分の濃淡が重ねられる。影の領域の計算の詳細は 2.1.1 節で述べる。

2.1.1 改良シャドウマップ法

影の領域の計算には、光源を視点とみなした場合のZバッファを利用した、いわゆるシャドウマップ法[8]を改良した方法を用いている。シャドウマップ法の特徴は、計算は速いが、本来は影でない部分が影と判定されたり、影の境界のエリアシングが激しいことである。前者の欠点の主な原因は、Zバッファを用いて離散的に奥行き値を保持しているために、面の法線ベクトルと光源ベクトルのなす角が90度に近くなると離散化誤差が大きくなることにある。その影響が小さくなるように改良を加えた。

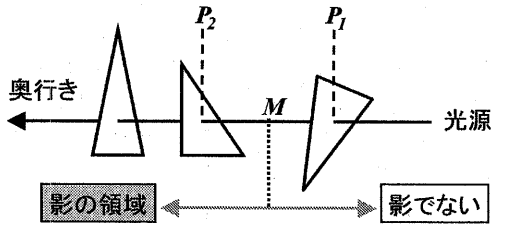

図2: 改良シャドウマップ法の説明図

改良法では、まずバッファを2枚用意し、一方は通常のシャドウマップ(各画素には光源から見て一番手前にある面の奥行き情報を保持)として、他方は、光源から二番目に手前にある面の奥行き情報を保持するバッファとする。前者を *Buffer1*、後者を *Buffer2* とすると、図2の例では、P_1 の値は *Buffer1* に、P_2 の値は *Buffer2* に保存される。

そして、影の領域か否かの判定の際には、判定点の光源からの奥行き値を d とした時、従来のシャドウマップ法が d と P_1 の値との比較を行うのに対して、改良法では、d を P_1 と P_2 の中点 M の値と比較を行い、M の値より小さければ、影の部分ではなく、M の値より大きければ、影の領域に含まれる(図2下部)としている。これにより、本来は影でない部分が影と判定されるエラーを軽減できる。

2.2 ストロークの生成

2.1節で生成した3つの画像の内の **tone-image** を目標画像とし、生成された線描画像をぼかした画像(**blurred-images** と呼ぶ)がそれに近づくように、以下のプロセスを繰り返し、ストロークのリストを生成する。ここで、リストには、ストロークの情報として、中心点 (C) と両端点 (P_0,P_2) の位置(図3参照)が保存され、それらからストロークを表すベジエ曲線の制御点が計算される。各ストロークをベジエ曲線で表現するのは、方向場(**direction-image**)に沿ったものとするためである。

2.2.1 置く位置と両端点の計算

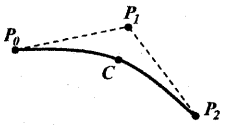

図3: ストローク

新たにストロークを描く候補点として、目標画像と **blurred-image** を比較し、最も描き込みの足りないと思われる画素を選択する。その際、ストロークの偏りを防ぐため、よりストロークが疎な領域の画素を選ぶ。次に、候補点を中心 (C) にして、方向画像の値(角度)に応じて両側に点を移動させ、ストロークの両端点 (P_0,P_2) を計算する。その際、**silhouette-image** を参照し、ストロークが輪郭や稜線と交わらないようにする。そして、曲線が C を通るように P_1 の位置を計算し、P_0,P_1,P_2 を制御点とする2次ベジエ曲線を判定のためのストロークとする。

2.2.2 フィルタをかけた画像の生成

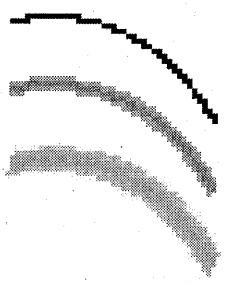

図4: 元のストローク(上)とそれをぼかした画像(中、下)

ストロークが生成された後、そのストロークにフィルタをかけてぼかした画像の生成を行う。この画像は、次節の密度の判定のために用いる。両端点を結ぶ線分の方向をストロークの方向とし、それに垂直な方向にストロークの各点をぼかすことでそれを行っている(図4)。**blurred-image** は生成中の線描画像をその都度ぼかすのではなく、このストロークをぼかした画像の重ね合わせて更新される。また、ストロークをぼかすためのフィルタは、ストロークの中心点でのストローク生成時点での **tone-image** と **blurred-image** の値の差に応じて変化させている。値の差が大きく、まだ描き込みの足りない部分では、あまりぼかさず(図4中)、値の差が小さく、もうあまりストロークを追加できない部分では、ぼかす幅を広めにしている(同下)。これはストロークの偏りを軽減し、**blurred-image** をより目標画像に近づけるためである。

2.2.3 ストローク密度の判定

blurred-image と目標画像である **tone-image** とを比較し、描き込み過ぎの部分がある場合には、このストロークは破棄され、描画されない。そうでなければ、ストロークのリストに加え、**blurred-image** を更新する。(実際にはストロークを中心点から伸ばしながら、同時に描き込み過ぎの判定も行っている。そのため、もうストロークが伸ばせなくなった時点で、ストロークの長さを調べ、それが閾値未満である場合に、そのストロークを破棄する、としている。これは主に、計算の効率化のための手法であるが、このようにしないと、(意図的に乱数を用いない限り) 一定の長さのストロークしか生成されず、また、ストロークの偏りのために、ストロークが密な部分にも“穴”が生じてしまう。)

新たなストロークが生成されなくなるまで (**tone-image** と **blurred-image** の誤差が閾値以下になるまで)、上記のストローク生成プロセス (2.2.1 ～ 2.2.3 節) を繰り返し、ストロークのリストに加えていく。

2.3 スクリーンへの描画

最後に、生成されたストロークと輪郭・稜線のスクリーンへの描画を行うのだが、この時、手描き風の感じを与えるためと、物体表面のテクスチャを表現するために揺らぎを付加して描画する (ストロークの描画方法の違いによる生成画像の変化については、図 10 を参照のこと)。

手描き風の揺らぎ

密度の判定 (2.2.3 節参照) の際には、ストロークを表すベジエ曲線は 2 次であったが、スクリーンへの描画時には次数を上げて 3 次とし、その制御点を変位させることで、ストロークに揺らぎを与える。ここで次数を変えているのは、判定時のコストは抑えつつ、描画時の揺らぎにはより変化を持たせるためである。また、輪郭や稜線についても同様にして揺らぎを与え、画面に描画する。

テクスチャの表現

物体表面のテクスチャを表現する方法として、ストロークの描画の際に以下の操作を施す。

1. 通常のテクスチャマッピングを行う。
2. テクスチャ画像の明度の値を高さとするハイトフィールドを考え、各画素毎に勾配ベクトルを計算し、二次元ベクトル場を作成する。
3. ストローク描画時に、そのベクトル場に従って、ストロークの各点を変位させる。

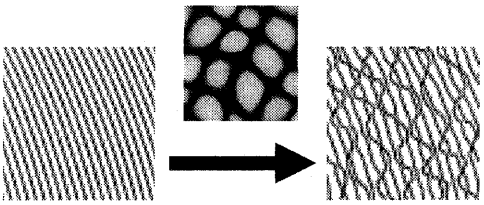

図 5: テクスチャを考慮したストロークの例

この操作を説明したものが図 5 で、操作を施す前の元のストローク群 (左) と、表面のテクスチャ (中央上)、施した後のストローク群 (右) の例である。

この方法を採用している理由は、物体形状に沿ったストローク (変位させる前の状態) の特徴も生かしつつ、テクスチャの表現を行うためである。

3 アニメーション化

ペンアンドインク風画像のアニメーションを作成するに当たり、考慮すべきこととして、以下の点を考える。

- アニメーションの前後のフレーム間でストロークが相関関係を持つ。
- ちらつきの原因となるストロークの急激な変化を抑える。
- 物体の移動に応じて、ストロークも移動する。
- 各フレームで物体の濃淡がきちんと表現されている。

これらの要求を満たすためには、2 節で説明した画像の生成法は適していると思われる。比較のために、他のペンアンドインク風画像の生成手法を考えると、例えば、物体表面の濃淡に応じたストローク密度のテクスチャを貼り付けることで画像の生成を行うものがあるが、これは上記の要求を満たすことが難しい。物体の濃淡の変化に伴いテクスチャを切り替えることが必要であったり、物体が拡大・縮小移動を行う際には面と面の境界部分に変化が偏ってしまうためである。一方、提案法では、ストロークは物体表面の濃淡値の変化に対して柔軟であり、また、視点と物体との距離には非依存にストロークが生成されるので、そのような問題は生じにくい。

しかしながら、一枚一枚の画像を独立に生成したのでは上に挙げた要求は満たされないので、前のフ

レームのストロークの保存を考える。そのために、保持されるストロークの情報の追加と、描画アルゴリズムの変更を以下のように行う。

保持されるストロークの情報

- 置く位置（物体の動きに伴って移動）
- 長さ（長さの変化は小さく抑える）
- 揺らぎ具合（揺らぎのシード値として保持）
- 消滅時間（過度に残るストロークを抑制）
- 濃さ（生成時や消滅時には徐々に変化させる）

輪郭・稜線についても揺らぎにフレーム間の相関を持たせるため、物体の各頂点にシード値が割り当てられており、それにより揺らぎが与えられる。

各フレームの描画アルゴリズム

1. 最初のフレームは、2節で述べた手法で画像を生成する。2番目以降のフレームの場合には、次のステップへ進む。
2. 前のフレームのストロークリストに含まれるストロークの位置を物体の動きに伴い移動させる。
3. 前のフレームのストロークから優先的に、次に生成されるフレームに描く。その際、通常の画像生成での条件に加え、見えない位置に移動したり、消滅時間に達したり、長さの変化が大きいストロークも破棄する。
4. 通常の画像生成と同様に、描き足りない部分に新たに追加のストロークを描く。

また、提案法では、階層構造を持った物体群を扱うことができ、キーフレームを指定することで、親ノードに対する平行移動と任意軸の回転移動を行うことができる。そのため、同様のデータ形式であるVRML等の既存のアニメーションデータを読み込んで描画を行うことも可能である。

紙面の都合上、アニメーションの例を示すことはできないが、以上のようにして生成されるアニメーションの特徴を挙げると、まず、利点として、物体が移動しても、ストロークが途切れてしまうことがなく、形状に沿って移動すること、常に（拡大縮小移動の場合にも）物体の濃淡をほぼ正確に表現していること、各フレームの生成時間が単独の画像一枚の生成時間よりも短いことなどある。しかし、現在の実装では、変化の少ない部分ほど、ちらつきが目立ってしまうという問題がまだ残っている。

4 結果とまとめ

提案法により生成された画像の例を図6〜9に示す。web上での実行を可能とするために、使用言語にはJavaを用い、アプレットとして実装されている。結果の描画面のサイズは640×480、描画時間の計測はPentiumIII 800MHzで行った(表1参照)。

生成画像	時間(秒)	ストローク数
トリケラトプス（図6）	12.2	2092
ボート（図7）	11.5	1717
頭蓋骨（図8）	9.0	1299
テーブルと家具（図9）	23.1	4544

表1: 描画時間とストロークの本数

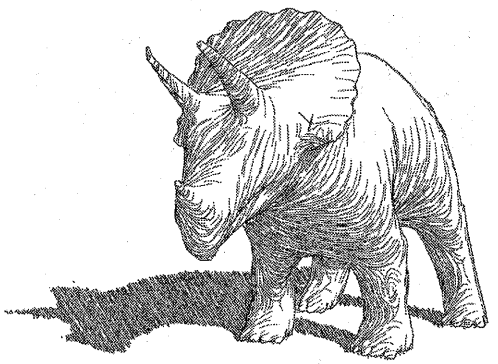

図6: 丸みを帯びた表面形状の例（トリケラトプス）

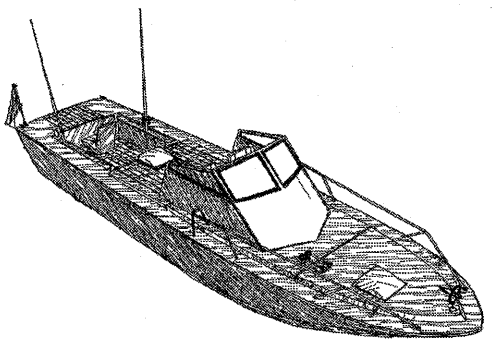

図7: 平らな面の多い場合の例（ボート）

次に、ストロークの描画方法の違いによる生成画像の違いを図10に示す。左上が通常のパラメータの例で、右上が揺らぎの大きさを10倍にした例、右下がストロークの太さを2倍にした例である。ま

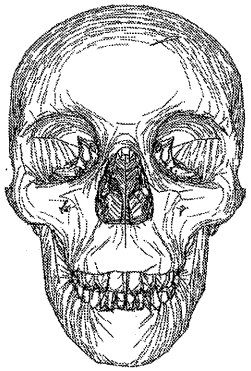

図 8: ポリゴン数の多い場合の例（頭蓋骨）

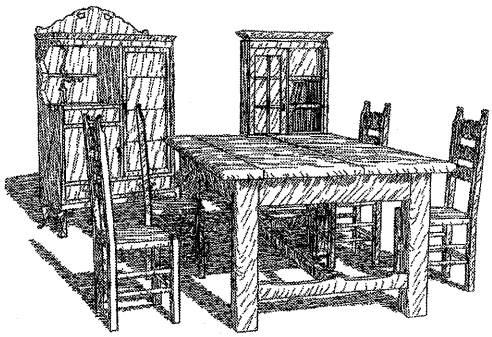

図 9: 複雑な物体の場合の例（テーブルと家具）

た、左下は、表面のテクスチャを考慮した場合の例（2.3 節参照）である。

本手法では、ユーザーがインタラクティブに操作し、各種のパラメータを設定することで、ストロークの密度や揺らぎ具合を簡単に変化させることができる。また、結果の生成画像は、影を考慮した物体の濃淡や形状（特に、丸みを帯びた物体（図 6、8）の形状）をよく表現しているものと思われ、描画に要する時間もインタラクティブに操作を行うのに耐え得る範囲であった。

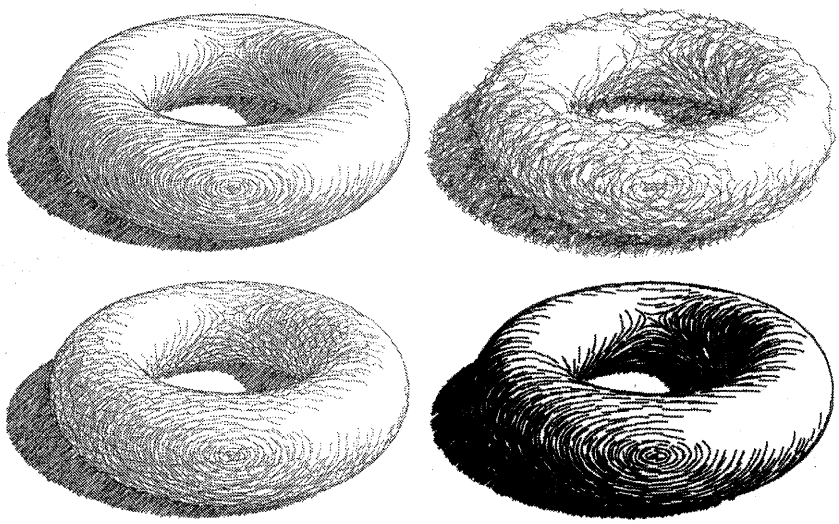

図 10: ストロークの描画方法の違いによる生成画像の違いの例

今後の課題

今後の課題としては、より自然なストロークの方向場の生成を行うこと、計算の効率化により、リアルタイムでの表示の実現などが挙げられる。

参考文献

[1] M.P. Salisbury, M.T. Wong, J.F. Hughes, and D.H. Salesin. Orientable Textures for Image-Based Pen-and-Ink Illustration. Proc. of SIGGRAPH '97, pp.401-406, 1997.

[2] L. Markosian, M.A. Kowalski, S.J Trychin, L.D. Bourdev, D. Goldstein, and J.F. Hughes. Real-time nonphotorealistic rendering. Proc. of SIGGRAPH '97, pp.415-420, 1997.

[3] T. Saito and T. Takahashi. Comprehensible Rendering of 3-D Shapes. Proc. of SIGGRAPH '90, pp.197-206, 1990.

[4] A.Hertzmann and D.Zorin. Illustrating smooth surfaces. Proc. of SIGGRAPH 2000, pp.517-526, 2000.

[5] M.P. Salisbury, S.E. Anderson, R. Barzel, and D.H. Salesin. Interactive Pen-and-Ink Illustration. Proc. of SIGGRAPH '94, pp.101-108, 1994.

[6] M.A. Kowalski, L. Markosian, J.D. Northrup, L. Bourdev, R. Barzel, L.S. Holden, and J.F. Hughes. Art-Based Rendering of Fur, Grass, and Trees. Proc. of SIGGRAPH '99, pp.433-438, 1999.

[7] G. Winkenbach and D.H. Salesin. Rendering Parametric Surface in Pen and Ink. Proc. of SIGGRAPH '96, pp.469-476, 1996.

[8] T. Möller and E. Haines. *Real-Time Rendering*, A K Peters, Ltd., pp.179-183, 1999.