グラフィクスとCAD 99－7
（2000．4．28）

Line描画におけるView-Dependentな Rendering手法に関する検討

張 英夏† 齋藤 豪‡ 高橋 裕樹† 中嶋 正之†

† 東京工業大学 情報理工学研究科,

‡ 東京工業大学 精密工学研究所

{chang, suguru, rocky, nakajima}@img.cs.titech.ac.jp

手描きのイラストや漫画などではより効果的な画像を生成するために特徴的な線を描いている．これまでにさまざまな特徴的線描画に関する手法が提案されているが，ユーザが自由に自分の嗜好に合った特徴的な線描画をすることは困難である．本研究ではより手描き風な画像を生成するために予めデータの各線に視線方向に応じたレンダリング情報をもたせ，レンダリング時にこれを参照するといった手法を提案する．この手法を用いることにより，ユーザがレンダリング情報を自由に登録することによってユーザの嗜好に合うレンダリングを実現させることが可能であり，また視線角度によって異なる太さでオブジェクトが描画されるように太さ情報を設定することによって，より微妙な線による手描き風なレンダリングが可能となる．

A Study for View-Dependent Line Rendering

Youngha Chang† Suguru Saito‡ Hiroki Takahashi† Masayuki Nakajima†

†Graduate School of Information Science and Engineering, Tokyo Institute of Technology.

‡Precision and Intelligence Laboratory, Tokyo Institute of Technology.

{chang, suguru, rocky, nakajima}@img.cs.titech.ac.jp

Abstract

In this paper, we propose a view-dependent rendering method for producing more hand drawing type images. For this, we attach some view-dependent rendering information to each line data in advance, and whenever we render model, we will utilize this information. By using this method, user can realize the following things: First, by registering the line thickness information freely user can make the result image according to user's preference. Second, by registering the thickness information differently according to the viewing angle, one can make more hand drawing type rendering images.

1. はじめに

手描きのイラストや漫画などでは線の太さを変えることによって線を強調したり省略したりストロークに特徴をつけている. このように線の太さを変えることによって内容を伝えやすくしたりある部分に視線を集中させることができる. しかし, 現在のコンピュータアニメーション制作手法では, 輪郭線や内形線にこのような特徴を反映させることができない.

そこで本研究では手描き風な画像を得るための方法として, 視線方向や描画する人の意図によって線の太さを自由に変化させることを可能にする VDLR(View-Dependent Line Rendering) 法を提案する. この手法では予め 3 次元モデルの個々の線に視覚方向に応じた描かれ方の情報をもたせ, レンダリング時に参照することにより, 視覚方向に応じて線の太さに変化をつけることが可能となる.

2. 線に着目した従来研究

ここでは線のレンダリングに関する従来研究についてまとめる.

従来の輪郭線生成法としては, 物体の色や明るさの不連続性を認識し, 線を発生させる方法であるエッジ検出法や対象物体の奥行き情報をもつ Z-buffer を利用し, 3 次元物体の境界を形成するポリゴンのふちに線を出させる方法である Z-buffer 法 [1] などがある. これらの輪郭線描画法では強調や省略したい輪郭線の描画や視線方向に対応じて異なった輪郭線を描画するなど自由な特徴付けを行うことは非常に困難である.

テクスチャを用いた線描画の従来研究として, 予めストロークテクスチャを定義し, ストロークテクスチャを貼ることによってより手描き風な画像を生成する方法 [2]-[4] や, 各画像に対しテクスチャと各点でのトーンやテクスチャの貼り付け方向を定義し, それらによって画像を生成する方法 [5], 筆の経路を決めそれに沿ってストロークを変形させてから貼るという手法 [7] などが提案されている. また, Appel's Algorithm[11] を向上させたアルゴリズムによって輪郭線を得てそれをレンダリングしていくという方法 [6] によりリアルタイムにレンダリングを行うという研究もなされている.

また, アニメーションにおける手描き風な描画に関する手法としては, 3 次元モデルを 2 次元画像に射影した後にストロークテクスチャを貼ることにより特徴的な線を描く方法 [8] が提案されている. その他にも, シルエット法 [1] が提案されている. シルエット法とは, 元のオブジェクトを少しだけ膨張させ, そこから元のオブジェクトを切り抜くことにより太さの異なるふちとり線を得る手法である. しかし, シルエット法では内形線は描画できない.

最後に線を強調してレンダリングを行う手法として, ノーマルベクトルなど幾何学的な属性を保存した G-buffer をいくつか用意し, それらによって得られた画像を合成することにより様々な線の強調結果を出せるようにした手法 [9] がある. また, 予め定義された強調部分や強調方法に関するいくつかのレンダリング規則により形状特徴を誇張するする手法 [10] が提案されている. ただし元々自動的に特徴線を描くことを目標としているので, ユーザが自分の嗜好に合った特徴の誇張方法を選択することはできない.

そこで本研究ではユーザにより多くの自由度を与える事により, より手描き風な画像生成を可能とする VDLR 法を提案する.

3. 提案 VDLR 法

本提案手法では線の太さを自由に変化させることを可能にするためにオブジェクトの各線に対しその線が各視線方向からどの程度の太さで見えるのかに関する情報を予め保存しておく. 後のレンダリング時にはこの情報にアクセスしてレンダリングの太さを決定しそれに従って線の描画を行う. この情報テーブルへのアクセスの方法としては線と視線方向ベクトルの関係を利用する.

第3.1節ではデータ形式について，第3.2-3.4節ではレンダリングアルゴリズムを順に説明する．

3.1 データ形式

前述のように各線をレンダリングするためには線の各視線方向でのレンダリング情報を保存したレンダリング情報テーブルが必要となる．従って，VRMLのような既存のデータ形式を本アルゴリズムで利用する場合，そのデータと共にそのオブジェクトの各線の描画情報を予め付加する必要がある．

そこでレンダリング情報テーブルという2次元配列を定義し，各視線方向からの線のレンダリングの太さtを保存する．ここでもしその視線方向において線が不可視な場合には-1を保存する．

3.2 描かれる候補線の決定

レンダリングのためにはまずオブジェクトが持つ全ての線に対し，各々の線が描画される可能性があるかを判断することによって描かれる描画候補線を決定する必要がある．

本手法の目的はユーザにより多くの自由度を与えることであるので，描かれる候補線としてはその線を含む2つの面のうち一つだけ可視である輪郭線と，二つの面が共に可視である内形線の全てを登録する．

ここで(1)のように定義した時，面Fの可視，不可視は(2)のように求められる．

$$\begin{cases} F : & Face\ of\ Polygon \\ V : & Vertex\ of\ FaceF \\ N : & Normal\ Vector\ of\ Face\ F \\ C : & Camera\ Position \\ E : & C - V \end{cases} \quad (1)$$

$$F = \begin{cases} Visible & if\ N \cdot E \geq 0 \\ Invisible & otherwise \end{cases} \quad (2)$$

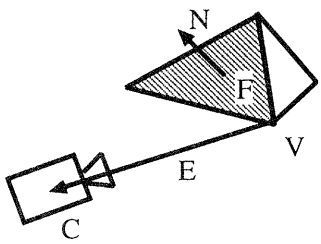
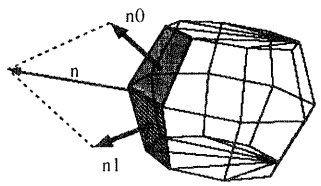


図 1: 面の可視/不可視の判定



n0,n1 : Face Normal
n : n0 + n1, Line Normal

図 2: Line Normalの定義

3.3 線のレンダリング太さの決定

前述の方法でレンダリングの候補線を決めた後にそれらをどの程度の太さでレンダリングするかを決定するため，レンダリング情報テーブルにアクセスする必要がある．このテーブルは各線と視線方向の関係によってアクセスすることが可能である．線と視線方向の関係は次のように求める．

まず(3)のように定義する．ここで，ラインベクトルlはxの値がプラスになるように定義し，ラインノーマルベクトルnはその線を含む隣接した二つの面のノーマルベクトル$n0, n1$を用い，図2のように定義する．また，oはlとnの外積を取ったものである．

次に，lをx軸，nをy軸，oをz軸と決めることにより，線に関する局所座標系を生成する．この時，この局所座標系の原点はlの中心点とする．

$$\begin{cases} l = (a_{11}, a_{21}, a_{31}) : line\ vector \\ n = (a_{12}, a_{22}, a_{32}) : line\ normal\ vector \\ o = (a_{13}, a_{23}, a_{33}) : l \otimes n \end{cases} \tag{3}$$

そして, この生成された線の局所座標系における視線ベクトル C_l の値を求める.

$$\begin{cases} C = (x, y, z) \\ \quad Camera\ Position \\ C_l = (x_l, y_l, z_l) \\ \quad Camera\ Position\ in\ Local \\ \quad Coordinate\ System \\ O_l = (0, 0, 0) \\ \quad The\ Origin\ in\ Local \\ \quad Coordinate\ System \\ A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \end{cases} \tag{4}$$

(4) のように定義した時, C_l はクラメルの公式を用い, (6)-(8) のように求められる.

$$detA = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \tag{5}$$

$$x_l = \frac{1}{detA} \times \begin{vmatrix} x - x_0 & a_{12} & a_{13} \\ y - y_0 & a_{22} & a_{23} \\ z - z_0 & a_{32} & a_{33} \end{vmatrix} \tag{6}$$

$$y_l = \frac{1}{detA} \times \begin{vmatrix} a_{12} & x - x_0 & a_{13} \\ a_{22} & y - y_0 & a_{23} \\ a_{32} & z - z_0 & a_{33} \end{vmatrix} \tag{7}$$

$$z_l = \frac{1}{detA} \times \begin{vmatrix} a_{12} & a_{13} & x - x_0 \\ a_{22} & a_{23} & y - y_0 \\ a_{32} & a_{33} & z - z_0 \end{vmatrix} \tag{8}$$

このような方法により線と視線方向の関係を表す C_l を得る. しかし, この C_l 自体はまだ変数を3つ持っている. レンダリング情報テーブルに迅速にアクセスできるように線と視線方向の関係をより少ない情報で表現し, 且つ後のレンダリング時や補間の際により簡単にデータにアクセスできるように, 直交座標系で表されている C_l を極座標系に変換する.

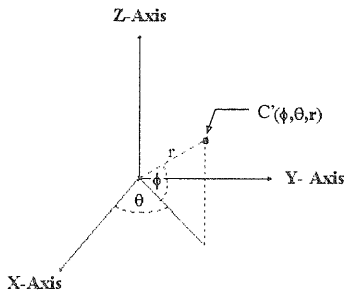


図 3: 極座標系

図3に示すように, 極座標系での3次元空間の各点は (ϕ, θ, r) の三つの変数を用いて表現されるが, ベクトルに対し正規化を行うとすると, 常にrは1となるので極座標系の3次元空間上の全ての点は二つの変数 (ϕ, θ) によって表すことが可能である.

直交座標系上の値 (x_l, y_l, z_l) から (ϕ, θ) への変換は次のように計算できる.

$$\phi = sin^{-1}(z_l) \tag{9}$$

$$\theta = \begin{cases} 0 & , \phi = \frac{\pi}{2}\ or\ \frac{3}{2}\pi \\ sin^{-1}(\frac{y_l}{cos(\phi)}) & , otherwise \end{cases} \tag{10}$$

こうして決定された (ϕ, θ) を利用して太さ情報テーブルにアクセスし, それによって線のレンダリング太さを決定する.

3.4 線のレンダリング太さの補間

レンダリング情報テーブル rIn へのアクセスがあった時, もしアクセス値 $(\phi, \theta) = (a, b)$ において太さ情報が定義されてない場合には定義されている太さ情報を用いて太さの推定を行う必要がある. x と $y(x < a < y)$ が ϕ において定義されていて z と $w(z < b < w)$ が θ において定義されている場合, (11)-(13) のような計算を用いて太さを推定する.

しかし, もし太さ情報に-1が入っているのなら補間せずに定義されているもう片方の太さ情報を用いることにする.

まず, $\phi = a$, $\theta = z$ での補間値を求める.

$$rIn[a][z] = \frac{(a-x)rIn[y][z] + (y-a)rIn[x][z]}{(y-x)} \tag{11}$$

次に, $\phi = a$, $\theta = w$ での補間値を求める.

$$rIn[a][w] = \frac{(a-x)rIn[y][w] + (y-a)rIn[x][w]}{(y-x)} \tag{12}$$

そして, このふたつを利用して, $\phi = a$, $\theta = b$ での補間値を求める.

$$rIn[a][b] = \frac{(b-z)rIn[a][w] + (w-b)rIn[a][z]}{(w-z)} \tag{13}$$

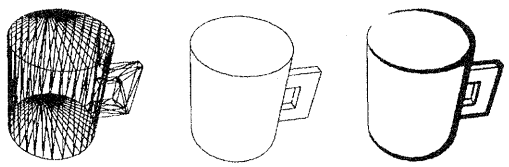
4. 実験結果

提案手法による結果を図 4-6 に示す.

図 4 では (a) が入力 3 次元ワイアフレームモデル, (b) が Z-buffer 微分法による一般的な輪郭線の描画結果, (c) が提案手法による輪郭線の描画結果である. 図で分かるように本手法では線の太さの強弱が自由に行え, 従来の手法より手描き風な輪郭線の描画が可能となることが分かる.

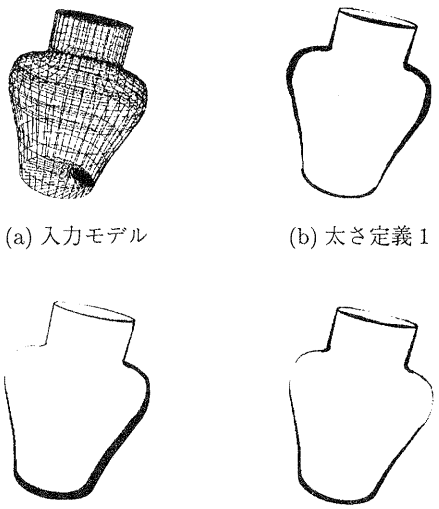
図 5 では壷に関し太さ情報テーブルの値を変えることによって (b) から (d) のように様々な太さでレンダリングした結果を示している. 本手法ではユーザが自分の好みに合わせ, 自由に輪郭線の太さ設定を行うことにより, ユーザの個性を表すような描画が可能であることが分かる.

図 6 ではりんごを描画した線の画像を紙のテクスチャーに合成したものである. 線に強弱が付いているため, 単純な形状のモデルであり且つ単純な配置であるにもかかわらず, 画面が貧弱になっていないことがわかる.



(a) 入力モデル (b)Z-buffer 微分法 (c) 提案手法

図 4: コップの描画比較



(a) 入力モデル (b) 太さ定義 1

(c) 太さ定義 2 (d) 太さ定義 3

図 5: 壷の様々な描画結果

5. おわりに

本研究では予め線のレンダリング情報テーブルを作成し, レンダリング時にこれを参照すると

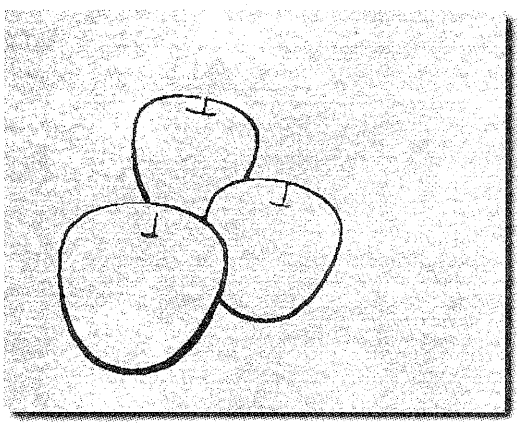


図 6: りんごの描画結果

いう方法を利用することによって作者の意図に応じて線の太さを自由に調整することを可能にした. また, 視線方向によって違う太さを与えることにより, view-dependent な線の描画をも可能にした.

しかし, 現在はまだレンダリング情報テーブル作成時に充分に user friendly な interface を作成してないため, 今後これに関して検討していく予定である.

また, 現在は線形的な補間しか行っていないため, 少しずつ回転し連続させた場合, スムーズに太さが変化していかない場合がある. 従って, より自然なアニメーションを制作するために補間法についても検討していく予定である.

参考文献

[1] 金子, 中嶋 : "セルアニメタッチ画像の生成のための 3 次元 CG 画像の 2 次元化アルゴリズム", 映像情報メディア学会誌, Vol.49, No.10, pp.1288-1295, 1995

[2] G.Winkenbach, D.H.Salesin : "Computer-Generated Pen-and-Ink Illustration", Proceedings of SIGGRAPH94, pp.91-100, 1994

[3] M.P.Salisbury, S.E.Anderson, R.Barzel, D.H.Salesin : "Interactive Pen-and-Ink Illustration", Proceedings of SIGGRAPH94, pp.101-108, 1994

[4] M.Salisbury, C.Anderson, D.Lischinski, D.H.Salesin : "Scale-dependet reproduction of pen-and-ink illustrations", Proceedings of SIGGRAPH96, pp.461-468, 1996

[5] M.P. Salisbury, M.T.Wong, J.F.Hughes, D.H.Salesin : "Orientable Textures for Image-Based Pen-and-Ink Illustration" , Proceedings of SIGGRAPH97 , pp.401-406, 1997

[6] L.Markosian, M.A.Kowalski, S.J.Trychin, L.D.Bourdev : "Real-Time Nonphotorealistic Rendering", Proceedings of SIGGRAPH97, pp.415-420, 1997

[7] S.C.HSU, I.H.H.Lee : "Drawing and Animation Using Skeletal Strokes", Proceedings of SIGGRAPH94, pp.109-118, 1994

[8] Kowalski, Markosian, Northrup, Bourder, Barzel, Holden, Hughes : "Art-Based Rendering of Fur, Grass and Trees", Proceedings of SIGGRAPH99, pp.433-438, 1999

[9] T.Saito,T.Takahashi : "Comprehensible Rendering of 3-D Shapes", Proceedings of SIGGRAPH90, pp.197-206, 1990

[10] 望月, 近藤, 佐藤, 島田 :"形状理解を容易にする特徴強調 画像の生成", 情報処理学会研究報告 95-CG-76,pp.73-80, 1995

[11] Appel : "The notion of quantitative invisibility and the machine rendering of solids", In Proceedings of ACM National Conference, pp.387-393, 1967