2009年度　卒　業　論　文

リアルタイム3DCGにおける輪郭線の誇張表現に関する研究

指導教員：三上 浩司 講師

メディア学部 ゲームサイエンスプロジェクト
学籍番号　M0106423
松尾 隆志

2009 年度　　卒　業　論　文　概　要

論文題目

リアルタイム 3DCG における
輪郭線の誇張表現に関する研究

メディア学部 **学籍番号 : M0106423**	**氏名**	松尾 隆志	**指導教員**	三上 浩司 講師

キーワード	3DCG, トゥーンレンダリング, 輪郭線, 誇張表現, リアルタイムレンダリング, ノンフォトリアリスティックレンダリング

近年、鉛筆や水彩画調などの手描き感のある画像を 3DCG を用いて再現する、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に行われている。その中に、漫画やディジタル 2D アニメーションのような表現を行うトゥーンレンダリングという手法があり、その特徴のひとつに輪郭線の表現がある。輪郭線の表現は、均一な太さの線で描く手法のほか、線の強弱や濃淡などで輪郭を描き、形状を強調する表現もある。これらの表現をリアルタイム 3DCG 上で行う方法として、絵画調などの表現の中で輪郭線を表現する研究や、無作為に線の太さを変えることで誇張した輪郭線を用いたゲームがある。しかし、絵画手法風の誇張した輪郭線は、ディジタル 2D アニメーション調の表現とは異なるものである。また、輪郭線や無作為に太さを変えるような輪郭線は、形状を効果的に表現しているとは言えない。

本研究ではデザインやデッサンの手法として用いる、形状を効果的に誇張表現する輪郭線の表現に注目した。この手法をもとに、リアルタイム 3DCG のトゥーンレンダリングおいて、輪郭をより効果的に表現する輪郭線の誇張表現手法を提案する。本手法では輪郭線の描画手法として、モデルを構成するポリゴンをすべて裏返したモデルである、裏ポリゴンモデルを用いた。裏ポリゴンモデルは、線が途切れることなく連続して表現でき、輪郭の表現が正確である。このため、本手法では裏ポリゴンモデルを利用した輪郭線表現手法を用い、誇張表現を行った。誇張表現を行う際にデザインやデッサンの手法に合わせ、対象となるモデルのポリゴン形状の弧を描くような部分を特徴とした。その特徴に従って裏ポリゴンモデルを変形することで、輪郭線の誇張表現を行った。

また、提案手法を実装したプログラムを用い、本手法を適用したものと、従来の表現を比較し効果を検証した。それぞれを比較した結果、通常の輪郭線表現よりも本手法の方が強弱のついた線の表現ができており、通常の均一な線や既存の無作為に線の太さを変える線よりも、モデルの凹凸を効果的に表現することができた。また、無作為に線の太さを変化する手法よりも、形状を考慮した誇張表現をすることができた。よって、本手法を用いることにより、リアルタイム 3DCG でのトゥーンレンダリングにおいて形状を効果的に表現する輪郭線の誇張表現を実現できた。

目 次

図 目 次

第 1 章

はじめに

1.1 研究背景

近年、鉛筆や水彩画調などといった手で描くような質感の画像を、3 次元コンピュータグラフィックス (以下、3DCG) を用いて再現するノンフォトリアリスティックレンダリング (Non-Photorealistic Rendering:NPR) の研究 [1][2] が盛んに行われている。その中で、漫画や 2D アニメーションのような表現をするトゥーンレンダリングという手法がある。トゥーンレンダリングは、数階調の陰影と細い輪郭線で表現しており、手描きのような質感を 3DCG 上で表現できる。特に日本では手描きと 3DCG を組み合わせる際に、それぞれを調和するためにトゥーンレンダリングを利用している [3]。本研究ではこのトゥーンレンダリングに用いる輪郭線に注目した。

1.2 問題提起

本研究で扱う輪郭線の表現を明確化するために、輪郭線の表現について説明する。人は物体を見るとき、明度や彩度、色彩などによって形を認識するが、輪郭や輪郭線を持つことでより簡単に形を認識できる [4]。そこで、本来物体にはない輪郭線を描いて意図的に形を強調したり、物体同士を区別する輪郭線を描いて差を強調したりする。この輪郭線はイラストや漫画、アニメーションなどのコンテン

ツで利用している。特に近年のディジタル 2D アニメーションにおいては、均一な輪郭線を多く利用している。これは、紙に作画した線をコンピュータ上にスキャンした際、コンピュータ上で均一な線として処理しているためである。また、3DCG 上でディジタル 2D アニメーション調の表現を行うトゥーンレンダリングでは、処理が最も単純である均一な線を多く用いている。しかし実際の作品では、イラストや漫画、水彩などの様々な線の表現を再現するために、あえて線の強弱や濃淡、点線や波線などで、輪郭を強調することがある。この部分的に強調する表現によって、対象とする物体を効果的に表現することが可能となり [5]、対象物をより豊かに表現できる。こうした表現をディジタル 2D アニメーションやトゥーンレンダリング上で行うために、フィルタワークや手作業によるレタッチなどの手間を加えることによって再現している。近年では、こうした輪郭線を誇張表現する作品の事例も増加している [6]。

本研究では、3DCG の表現においてゲームなどでの利用を想定し、リアルタイムレンダリングを用いる。リアルタイムレンダリングとは 1 秒間に 30〜60 回描画を行い、かつその描画速度を維持するという制約が生じる中で表現を行っていくものである。一方で、時間をかけてひとつずつ画像を生成するプリレンダリングという手法もある。この手法では描画速度の制約がないため、生成した画像に対して輪郭の検出や線の描画などの画像処理を行うことで、誇張した輪郭線を描画している。このプリレンダリングで行う手法は、時間や処理速度をかけ、精細な画像処理を行うことで可能な手法であり、リアルタイムレンダリングで同様の手法を行うことや同様の質を出すことは難しい。

“JET SET RADIO”[7] や “ドラゴンクエスト VIII”[8] などを始め、リアルタイムレンダリングでトゥーンレンダリングを行っている例は多くある。その多くが様々な手法を用いることで、均一な輪郭線の表現を行っている。NPR の分野でも、リアルタイムに絵画調などの表現をする研究 [9][10] があり、輪郭線の太さを変えるなどの誇張した表現を実現している。トゥーンレンダリングでの輪郭線表現にそれらの表現を用いた場合、線が絵画調や絵画手法に則したものとなる。このため、

ディジタル 2D アニメーションのような表現とはまったく異なったものとなる。また、リアルタイムでトゥーンレンダリングを行っている 3DCG ゲーム “大神”[11] では、誇張した輪郭線の表現を用いている。これは輪郭線を無作為に揺らすような手法で誇張した表現を行っているが [12]、形状を効果的には表現していない。このように、NPR での輪郭線表現ではディジタル 2D アニメーションのような表現の中で、形状を効果的に表現するものとなってはいない。同様に、既存のリアルタイムレンダリングでのトゥーンレンダリングにおける輪郭線の誇張表現も、形状を効果的に表現するものとなってはいない。

1.3 研究の目的

本研究では、リアルタイム 3DCG でのトゥーンレンダリングにおける輪郭線をより効果的に表現する誇張表現手法を提案する。1.2 節で述べたように、輪郭線の誇張表現には線の強弱や濃淡、波線や点線などの表現方法がある。その中で本研究では線の強弱による表現に注目した。線に強弱をつける手法は、線の太さを変えるだけのシンプルな手法であり、イラストや漫画などで多用している手法である。また、描く人の筆圧やストロークの変化が直接線として出る手法であり、弧を描く形状や奥行きを表現する際に用い、丸みや立体感や強調の効果を与える。このように線に強弱をつける手法は、形状を効果的に表現する手描きならではの手法である。本研究では特に、弧を描くような曲線となる部分を特徴として誇張した表現を行う。以上を踏まえ、本研究では効果的に誇張した表現を行うために、目的として次の 3 点を満たすことを想定し、手法を提案する。

- リアルタイム 3DCG でのトゥーンレンダリングで輪郭線の表現を行うこと。
- 輪郭線の太さの変化で 3D モデルの形状をより効果的に表現すること。
- 連続した強弱の変化で手描きに近い輪郭線の表現を行うこと。

本論文では、以上の想定を満たすリアルタイム 3DCG での輪郭線の誇張表現手法について提案する。また、本手法を実装したプログラムを用い、リアルタイム

3DCG 上で輪郭線の誇張表現の検証を行う。これにより、モデルのエフェクト効果や、モデラーによるモデルの演出のひとつとして利用できるようにする。

1.4 論文構成

本論文では、第 2 章でリアルタイム 3DCG での輪郭線の表現手法と本研究が目標とする表現について説明し、第 3 章では輪郭線の誇張表現についての具体的な手法を説明する。第 4 章で実際に生成した輪郭線の検証を行い、第 5 章では本手法の評価と現状における制約及び展望について述べる。第 6 章では、全体のまとめと今後の課題を述べる。

第 2 章

リアルタイム3DCGにおける輪郭線の表現

本章では、リアルタイム 3DCG における輪郭線の表現手法について説明するとともに、本研究で実現する輪郭線の誇張表現について明確化する。2.1 節ではポリゴンの面方向による輪郭線について、2.2 節では画像処理を利用した輪郭線について、2.3 節では 3D モデルを拡大しポリゴンを裏返した手法による輪郭線について述べる。最後に 2.4 節で本手法で目標とする輪郭線の誇張表現について述べる。また、リアルタイム 3DCG における輪郭線の表現手法について、“リアルタイムレンダリング 第 2 版”[13] で挙げている手法を参考にした。

2.1 ポリゴンの面方向による輪郭線

本節では、ポリゴンの面方向から輪郭線を描画する方法を説明する。ポリゴンの面方向とは、あるポリゴンに垂直なベクトルである、法線ベクトルが向いている方向とする。この法線ベクトルと視点の方向ベクトルを反転したベクトル (以下、「視点の逆方向ベクトル」) の内積の値から面の傾きを求め、輪郭を検出し、輪郭線を描画する。この手法は、Gooch ら [14] が提案し、Marshall はこの手法をリアルタイムで実行する方法を “Game Programing Gems 2”[15] で紹介している。

図 2.1 はポリゴンの面方向によって輪郭を検出した結果を描画したものである。

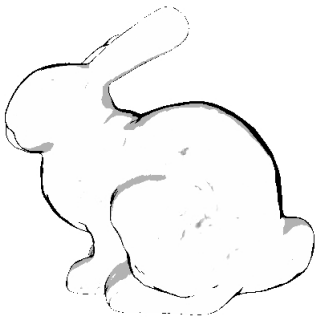
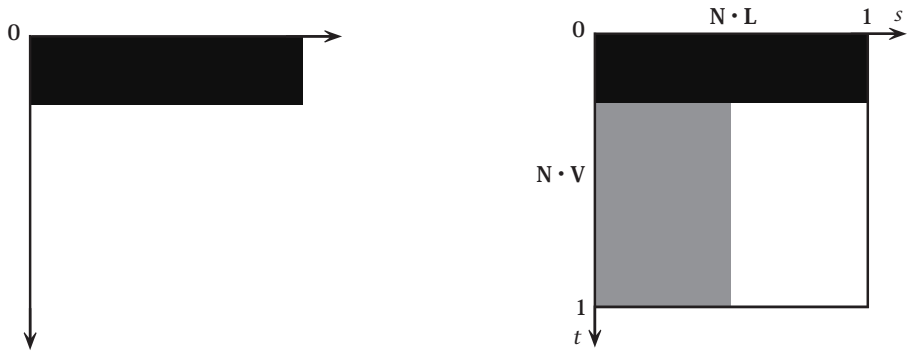


図 2.1: ポリゴンの面方向による輪郭線

ポリゴンの単位法線ベクトルと視点の逆方向単位ベクトルの内積の値が 1 に近い場合、ポリゴンの面が視点の方に向いている状態であり、表面であると判断する。逆に内積の値が 0 に近い場合、ポリゴンの面が視点から見て真横に近い状態であり、輪郭であると判断する。このポリゴンの単位法線ベクトルと視点の逆方向単位ベクトルの内積の計算を、画面上に表示しているポリゴンに対して行う。その結果の内積の値をテクスチャ座標に用い、0 に近い部分の色を変えた 1 次元テクスチャをポリゴンに貼り付けることで輪郭線を描画する。1 次元テクスチャとは、高さが 1 ピクセルである画像を持つテクスチャのことであり、このテクスチャを貼り付けるマッピングという作業を行うことで輪郭線を描画する。図 2.2(a) は、マッピングに用いる 1 次元テクスチャの例である。

図 2.2(b) は、1 次元テクスチャのマッピングに用いる図 2.2(a) を含んだ、2 次元テクスチャの例である。面の単位法線ベクトルを $\mathbf{N}$ 、光源の方向ベクトルを反転した単位ベクトル (以下、「光源の逆方向単位ベクトル」) を $\mathbf{L}$ とした時の明度を、面の単位法線ベクトルと光源の逆方向単位ベクトルの内積 $\mathbf{N} \cdot \mathbf{L}$ で求める。これが 2 次元テクスチャの s 方向の座標となる。また、視点の逆方向単位ベクトルを $\mathbf{V}$ とした時、面の単位法線ベクトルと視点の逆方向単位ベクトルの内積 $\mathbf{N} \cdot \mathbf{V}$ がテクスチャの t 方向の座標となる。そして s 方向、t 方向の座標を用い、t 座標が 0 に近く輪郭と判断できる場合、1 次元テクスチャのマッピングを行うことになり、

輪郭線が描画できる。また同様に s 方向、t 方向の座標を用い、t 座標が 1 に近く面であると判断できる場合、図 2.2(b) の灰色や白色部分をマッピングし、陰影を表現する。s 座標は明度を表すため、0 に近い場合は明度が低く、1 に近い場合は明度が高い。このことを利用し、灰色部分や白色部分を変えることで、様々な段階的な陰影の表現を行うことができる。



(a) 1 次元テクスチャの例　　(b) 2 次元テクスチャの例

図 2.2: テクスチャマッピングに用いるテクスチャの例

このポリゴンの面方向を利用した手法は、面を黒く塗ることで輪郭とみなしている。そのため、ポリゴンの面と面がなす角度によって描画する線の幅が変化する特徴がある。例えば、大きく平らなポリゴンの場合、視点から真横に近い状態の際には黒くなる。しかし、そのポリゴンの全てが黒く見えるために、望ましい結果にはならないことがある。図 2.3 は立方体のモデルを用い、複数の面が視点から真横に近い状態を表した図である。このように、面全体が黒くなっている状態は輪郭線とは言えない。

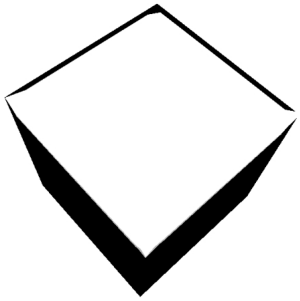


図 2.3: 面の全体が黒くなる例

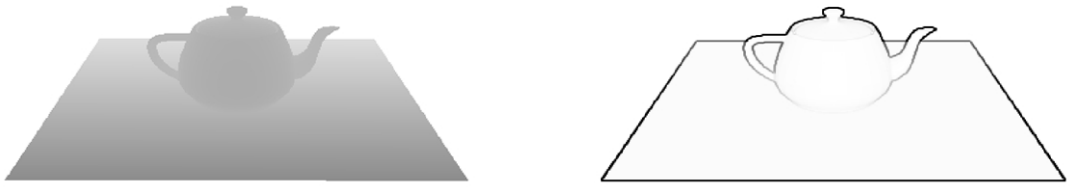
2.2 画像処理を利用した輪郭線

本節では、画像処理を利用して輪郭線を描画する手法を説明する。画像処理を利用する手法とは、3 次元のモデルを画像として描画し、その画像に対して画像処理を行うことで輪郭を抽出し、輪郭線を描画する手法である。この手法は、Saito ら [16] が元となる手法を提案し、Decaudin[17] がトゥーンレンダリングのために改良した。Decaudin の手法に限らず、画像処理で描画する手法は様々な種類があり、鉛筆調 [18] や水彩画調 [19] のレンダリングなど、トゥーンレンダリングではない表現でも利用している。Decaudin の手法は、デプス (深度) バッファを利用する手法と、法線バッファを利用する手法を組み合わせることで、輪郭線を描画している。この手法の他にも、マテリアルの境界に輪郭線を描画する手法や、モデルのシルエットから輪郭線を描画する手法などがある。本節ではスタンダードな Decaudin の手法を説明する。

まず、デプスバッファを利用する手法について説明する。デプスバッファとは、生成した画像の各ピクセルとポリゴンの距離から算出するデプス (深度) 値を格納するメモリ領域及び格納する技術のことである。この手法では、隣接するポリゴンのデプス値の不連続性に対し、画像処理を用いることで輪郭を検出し、輪郭線を描画する。

図 2.4 はデプスバッファによる輪郭の検出に関する画像を示したものである。図

2.4(a) は、デプス値をデプステクスチャに書き込み、ポリゴンにマッピングした際の画像である。このデプステクスチャを用いてマッピングした 3D モデルを描画した画像をもとに輪郭を検出する。輪郭を検出する画像処理の手法は様々あるが、多くは Laplacian フィルタや Sobel フィルタによる輪郭検出を用いる。Laplacian フィルタは画像に対して 2 次微分を、Sobel フィルタは画像に対して 1 次微分を行うことで輪郭を検出するフィルタであり、この検出結果から輪郭線を描画する。図 2.4(b) は、デプスバッファのマップである図 2.4(a) に Sobel フィルタの処理を施し、輪郭線を描画した画像である。



(a) デプステクスチャをマッピングした図　　(b) デプスバッファによる輪郭線

図 2.4: デプスバッファを利用した輪郭線検出

次に、法線バッファを利用する手法について説明する。法線バッファとは、ポリゴンから垂直に伸びる法線の情報を格納するメモリ領域及び格納する技術のことである。この手法では、3D モデルの形状データから法線の情報を取得し、凸凹を調べた結果に画像処理を加えることで輪郭を検出し、輪郭線を描画する。

図 2.5 は法線バッファによる輪郭の検出に関する画像を示したものである。

(a) 法線テクスチャをマッピングした図　　(b) 法線バッファによる輪郭線

図 2.5: 法線バッファを利用した輪郭線検出

図 2.5(a) は、ポリゴンの法線方向をテクスチャに書き込み、ポリゴンにマッピングした際の画像である。このマッピングした画像をもとに輪郭を検出する。輪郭を検出する画像処理の手法は、デプスバッファを利用する方法と同様に、Laplacian フィルタや Sobel フィルタによる輪郭検出を用いる。図 2.5(b) は、図 2.5(a) に Sobel フィルタの処理を施すことで輪郭を検出し、輪郭線を描画した画像である。

最後に、バッファ上にある図 2.4(b) のようなデプスバッファによる画像と、図 2.5(b) のような法線バッファによる画像を元の画面に乗算し合成することで、輪郭を画面に描画する。つまり、画像の輪郭をモデルの輪郭として扱うことで輪郭線を描画している。

画像処理を用いる手法をリアルタイムで行うために、Card ら [20] は、画像処理を専門とする補助演算装置である GPU(Graphics Processig Unit) 上で、モデルの頂点を扱うバーテックス (頂点) シェーダを利用する手法を提案した。この手法では、法線バッファは法線マップとして RGB で表す色チャンネルとして、デプスバッファはアルファチャンネルとして、それぞれテクスチャに書き込むことで輪郭検出の前段階となる処理を行っている。また、Mitchell ら [21] は Card らの手法

でレンダリングした画像から、GPU 上でピクセルを扱うフラグメント (ピクセル) シェーダを利用した輪郭検出フィルタで輪郭線を描画する手法を提案している。

画像処理を用いる手法の利点として、画像処理の方法によって様々な線を描画できる点が挙げることができる。デプスバッファを利用する手法と法線バッファを利用する手法では、同じ輪郭検出フィルタを使用しても違う結果となる。同様に、同じバッファを利用する手法でも、Laplacian フィルタや Sobel フィルタなど、利用する輪郭検出フィルタの種類によって検出する輪郭が変わり、違う結果となる。また、新たに別の画像処理を加えることで輪郭線の太さを変えるなど、様々な加工を追加することが可能である。

一方で、デプスバッファや法線バッファを利用する手法には、比較部分の差が小さい場合では違いを認識できない問題がある。例えば、机上にある 1 枚の紙をレンダリングする場合、デプスバッファの場合は机と紙のデプス値の差が小さいため認識できず、法線バッファの場合は机と紙の法線ベクトルが等しいため認識できない。このため、デプスバッファや法線バッファを用いる手法はモデルの輪郭を正確に描画しているとは言えない。また、高速に実行するための手法でもある、Card らや Mitchell らの手法を用いる場合には、ハードウェアである GPU が必要となる。

2.3 裏ポリゴンモデルを利用する輪郭線

本節では、裏ポリゴンモデルを利用した輪郭線の描画手法について説明する。裏ポリゴンとは、モデルを構成しているポリゴンの裏側にあたる部分のことである。また、裏ポリゴンモデルとはモデルを構成するポリゴンをすべて裏返したモデルであり、黒く塗ったポリゴンが内側を構成しているモデルでもある。本手法ではこの裏ポリゴンモデルを輪郭線とみなす。この手法は Rossignac ら [22] がモデルの背面を拡大することで、輪郭線を描画させる手法を最初に提案し、Evan ら [23] が裏ポリゴンモデルを利用する手法を提案した。

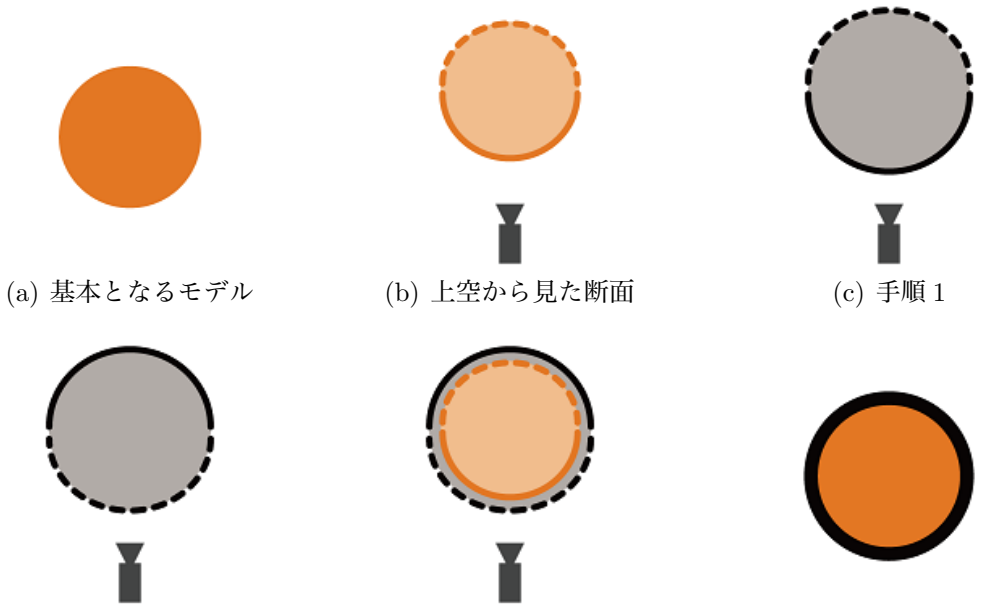
裏ポリゴンを利用した輪郭線の描画手法の概略として、鈴木の研究 [24] における、3D モデルを拡大し裏ポリゴンにして輪郭線にする手法を次に述べる。

1. 元となるモデルを複製した後、頂点の法線方向に拡大し、面の色を黒くする。頂点の法線とは、その頂点を構成要素に持つポリゴンの法線ベクトルの平均とする。

2. 拡大したモデルの表裏を反転する。このとき、視点から見て手前になるモデルの面は表示せず、奥となる面を表示する。

3. 拡大し、ポリゴンの表裏を反転したモデルに、元々のモデルを重ね合わせる。

図 2.6 は球体のモデルを利用して、3D モデルを拡大し裏ポリゴンにして輪郭線にする処理を図示したものである。



(a) 基本となるモデル　(b) 上空から見た断面　(c) 手順 1

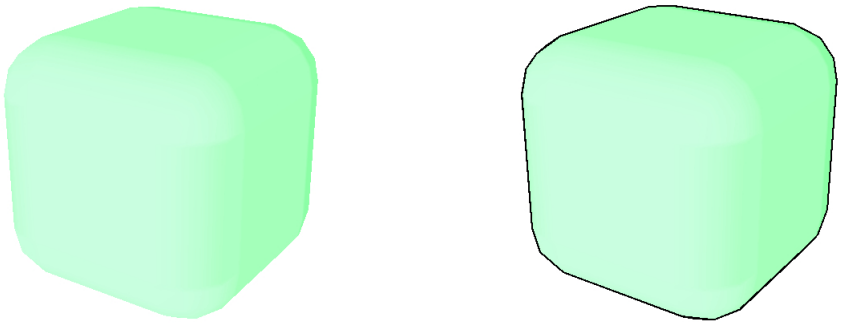
(d) 手順 2　(e) 手順 3　(f) 最終的なモデル

図 2.6: 裏ポリゴンモデルを利用した輪郭線描画の過程

図 2.6(a) は対象となる球体モデルを正面から見た図である。図 2.6(b)〜2.6(e) は、モデルを上空から見下ろした時の断面図であり、実際の視点は図の下側にあるも

のとする。また、実線部は視点から見て表示している面、点線部は表示していない面を表す。図 2.6(b) は元のモデルを上空から見た時の断面図であり、図 2.6(c) は複製して拡大し、面を黒く塗ったモデル、図 2.6(d) はポリゴンの表裏を反転したモデル、図 2.6(e) は元々のモデルを重ね合わせた図となっている。この処理を用いた結果、図 2.6(e) のように元のモデルからはみ出した部分が表示される。図 2.6(f) が実際の視点から見た図であり、球体のモデルに輪郭線を描いたように見える。

図 2.7 は、以上の処理を 3D モデルに適用した結果を示した図である。



(a) 元となるモデル　　(b) 輪郭線を描画したモデル

図 2.7: 裏ポリゴンを利用した輪郭線

元のモデルである図 2.7(a) に、処理を行った結果が図 2.7(b) であり、十分に輪郭線が描画できていることが分かる。この手法の特徴として、均一な太さの輪郭線が簡単な処理で描画できる点がある。そのため、細く一定な輪郭線の表現を行う際には有効であり、対象となるモデルに対して正確に輪郭線を描くことができる。

ただし、裏ポリゴンモデルを利用した輪郭線の描画手法は、元となるモデルと輪郭線用のモデルの 2 つが必要となるため、表示するモデルの頂点数やポリゴン数が倍になる問題がある。これは少ないポリゴン数であるとそれほど影響はないが、多くなると負荷がかかり描画速度が低下する。ただし、モデルの頂点を法線方向に動かすことで輪郭線を描画しているため、GPU 上で頂点を扱うバーテック

スシェーダを利用することで、負荷を軽減することは可能である。

また、立方体のようなモデルを真横から見た状態でモデルの頂点をそれぞれの法線方向に拡大し輪郭線を描画すると、角に隙間が生じる場合がある。立方体のようなモデルの角の頂点は見た目では 1 つしかないが、実際には 3 つの頂点が重なっている状態である。このため、頂点を法線方向に単純に拡大した場合はそれぞれの頂点が独立して移動するため、面が独立する可能性がある。

これらの問題に対し、他の手法より簡単かつ明確に均一な太さの輪郭線が描画できるため、この裏ポリゴンモデルを利用した輪郭線描画手法は、トゥーンレンダリングを用いたコンテンツの多くが利用している手法である。

2.4 目標とする輪郭線の誇張表現

本節に至るまで、リアルタイム 3DCG における輪郭線表現について述べた。

ポリゴンの面方向から輪郭線を描画する手法では、ポリゴンの面を輪郭とみなすことで輪郭線を描画した。これにはポリゴンの面方向によって太さが変わる特徴があり、太さが変わるような輪郭線の表現が可能である。しかし、視点から見て真横に近い状態では面がすべて黒く見える問題があり、輪郭線を正確に描画していない。一方、画像処理を利用する手法では、画像処理の内容次第で様々な線を描画できる特徴がある。この手法では、画像の輪郭の線をモデルの輪郭線として描画する。デプスバッファや法線バッファを利用する際には、比較する距離や法線方向が近い場合は違いを認識できないため、正確に輪郭線を描画することは難しい。また、裏ポリゴンモデルを利用して輪郭線を描画する手法では、背面となる裏ポリゴンモデルを輪郭として輪郭線を描画する。この手法では形状をそのまま拡大するため正確で均一な線が描画できるが、モデルが 2 つ存在することになるため、表示する頂点数やポリゴン数が倍になってしまう問題がある。また、頂点が別々の法線を持つ場合、角に隙間が生じる問題もある。ただし、大きな問題もなく正確な輪郭線が簡単に描画できるため、多用している手法である。

表 2.1 は、既存手法における輪郭線表現の特徴をまとめた表である。

表 2.1: 既存の輪郭表現手法における特徴

種類	輪郭の正確性	実行速度	線の制御	描画の種類
ポリゴンの面方向	×	○	×	面を輪郭とみなす
画像処理	×	△	△	画像の輪郭の線
裏ポリゴンモデル	○	△	○	背面のモデルの面

リアルタイム 3DCG にて輪郭線の誇張表現を行う本研究において重要なことは、「形状を効果的に表現をするための正確な輪郭線」と、「リアルタイムレンダリングを維持可能な実行速度」である。これに対し、ポリゴンの面方向による輪郭線表現手法の場合、高速に実行可能である。しかし、面を輪郭とみなす手法のため輪郭は正確に描画していない。このため、意図的に輪郭線の太さを変えるなどの制御は難しい。次に、画像処理を用いた輪郭線表現手法の場合、単純な処理の場合は高速に行うことができる。しかし、輪郭の検出や輪郭線の描画や制御などを精細にを行う場合、リアルタイム性を損なう。また先述の通り、輪郭が正確に描画できないため、輪郭線の制御は難しい。最後に、裏ポリゴンモデルを用いた輪郭線表現手法の場合、モデルの頂点数やポリゴン数が 2 倍に増加し、速度が低下する可能性がある。しかし、基本的には速度を維持しながら正確な輪郭線を描画できる。また、裏ポリゴンモデルを変形することで、輪郭線の太さを変えるなどの制御も可能である。このため、本研究ではリアルタイム性があり、正確な輪郭線を描画できる裏ポリゴンモデルを利用した輪郭線描画手法を用いる。この手法をもとに、任意に線の強弱をつけることが可能で、形状を誇張する輪郭線を描画する手法を新たに提案する。この誇張した輪郭線の表現を可能にすることで、モデルの形状をより効果的に表現することを可能にする。最後に、本研究の手法を適用したものと従来の表現を比較することで手法の効果を検証し、リアルタイム性の検証も行う。

第 3 章

輪郭線の誇張表現手法

本章では、3DCG でのトゥーンレンダリングにおける輪郭線を誇張して表現する手法を述べる。まず、3.1 節では誇張表現を行う対象である 3D モデルの特徴の検出手法について述べる。次に、3.2 節では 3.1 節で検出した特徴に従い輪郭線の誇張表現を行う手法について述べる。最後に、3.3 節で本手法を実装したプログラムの手順を示す。

また、本研究では 1.3 節で述べた通り、対象となるモデルの弧となる部分を特徴とした。このため、特徴である弧を描く部分を検出し、その結果に対して誇張した輪郭線を描く。また、誇張する輪郭線は無作為に揺らすような手法ではなく、連続した変化のある線にする。

3.1 特徴点の検出

誇張した表現を行うためにまず、特徴となるモデルの弧を描く部分を検出する。3DCG での形状にて弧を描く部分となるのは、モデルの凹形部分と凸形部分である。図 3.1 はその例を示したものであり、青い丸で囲んだ部分がモデルの凹形部分、赤い丸で囲んだ部分がモデルの凸形部分である。

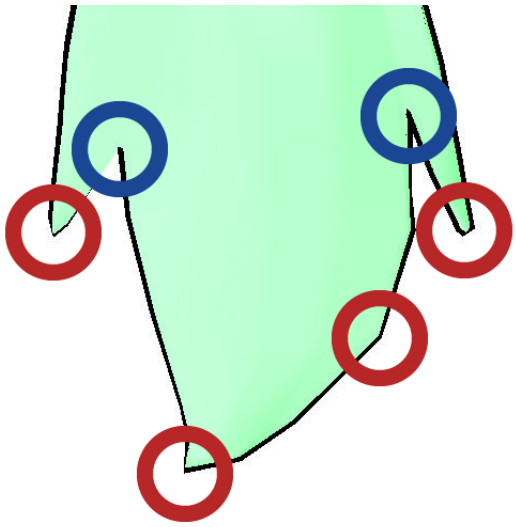


図 3.1: モデルの凹凸部分

図 3.1 の青い丸、赤い丸のような形状をする部分を検出するために、地神の研究 [25] における、各頂点の法線を利用した 3D モデルの凹凸部分を検出する手法を拡張して用いる。

図 3.2 は、ポリゴンに接続している頂点とその法線ベクトルを表したものである。

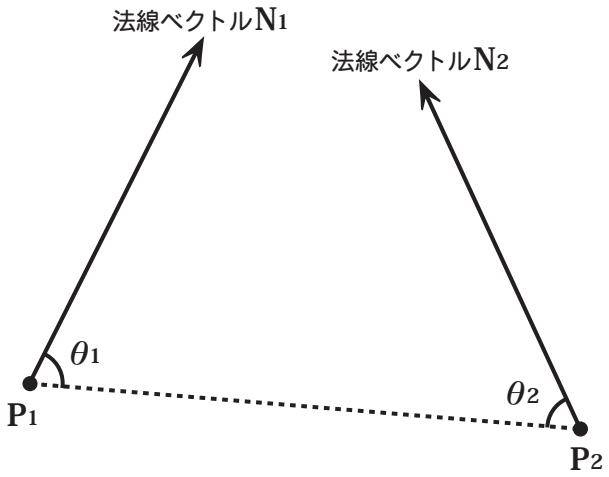


図 3.2: 接続している頂点とその法線ベクトル

モデル上のある頂点座標を $\mathbf{P_1}$ とし、その頂点に接続する頂点座標を $\mathbf{P_2}$ とする。

また、この頂点の法線ベクトルをそれぞれ $\mathbf{N_1}$、$\mathbf{N_2}$ とする。頂点の法線ベクトルとは、その頂点を構成要素に持つポリゴン面の法線ベクトルの平均とする。また、線分 $\mathbf{P}_1\mathbf{P}_2$ と $\mathbf{N_1}$ がなす角度を θ_1、線分 $\mathbf{P}_2\mathbf{P}_1$ と $\mathbf{N_2}$ がなす角度を θ_2 とする。

θ_1 と θ_2 が式 (3.1) を満たす場合、凹形状のポリゴンの接続である。

$$0^\circ \leq \theta_1 + \theta_2 < 180^\circ \tag{3.1}$$

図 3.3 は、法線ベクトルが向かい合っている、凹形状のポリゴンの接続を示した図である。

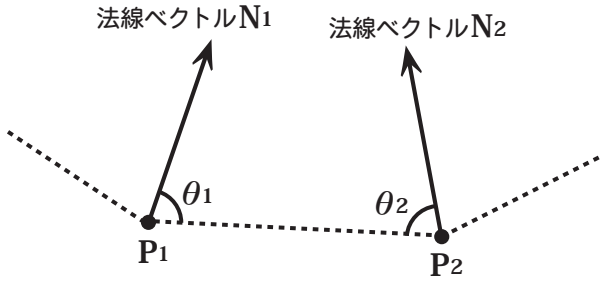


図 3.3: 凹形状のポリゴンの接続

この場合、θ_1 と θ_2 の合計が小さいほど、法線ベクトルがより向かい合っている状態である。全ての頂点において、各頂点の法線ベクトルとその頂点に隣接する頂点の法線ベクトルを比較し、式 (3.2) を満たす任意の角度 A 以下である、隣接する頂点の数を求める。

$$A > \theta_1 + \theta_2 \tag{3.2}$$

一方、θ_1 と θ_2 が式 (3.3) を満たす場合、凸形状のポリゴンの接続となる。

$$180^\circ \leq \theta_1 + \theta_2 < 360^\circ \tag{3.3}$$

図 3.4 は、法線ベクトルが反対を向いている、凸形状のポリゴンの接続を示した図である。

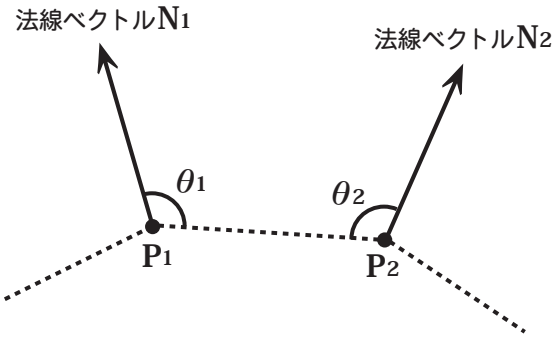


図 3.4: 凸形状のポリゴンの接続

この場合、θ_1 と θ_2 の合計が大きいほど、法線ベクトルがより反対を向いている状態となる。こちらも同様に、全ての頂点において各頂点の法線ベクトルと隣接する頂点の法線ベクトルを比較し、式 (3.4) を満たす任意の角度 B 以上である、隣接する頂点の数を求める。

$$B \ < \ \theta_1 + \theta_2 \tag{3.4}$$

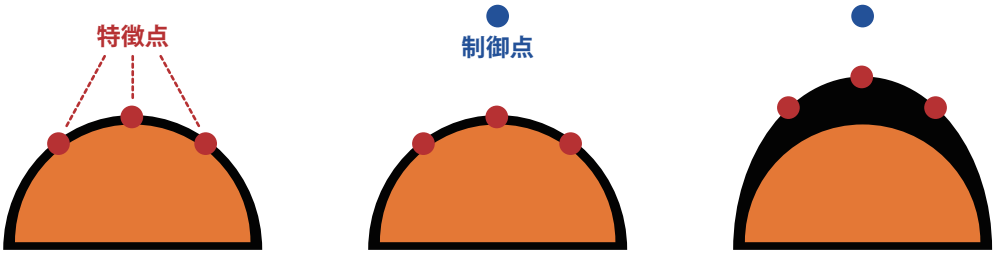
各頂点において、式 (3.2) を満たしている隣接する頂点数が多い場合、図 3.1 の青い丸で囲んだ部分であるとして、その頂点の情報を凹形状の特徴部分として保存しておく。同様に、式 (3.4) を満たしている隣接する頂点数が多い場合、図 3.1 の赤い丸で囲んだ部分であるとして、その頂点の情報を凸形状の特徴部分として保存しておく。

3.2　制御点を利用したモデルの変形

3.1 節で検出した特徴部分である頂点 (以後、特徴点と呼ぶ) を移動し、裏ポリゴンモデルを変形することで、輪郭線に強弱をつける。本研究の輪郭線の表現では、2.3 節の裏ポリゴンモデルを利用した輪郭線の表現手法を用い、次のような処理を行うことで、誇張した輪郭線の表現を行う。

1. 裏ポリゴンモデルを描画し、元のモデルの特徴点である頂点と互換する。
2. モデル周囲に変形を制御するための点 (以後、制御点と呼ぶ) を配置する。
3. 特徴点から最も近い制御点を割り出し、各特徴点と制御点を関連付ける。
4. 特徴点と制御点の距離に応じて特徴点を頂点の法線方向に移動する。

図 3.5 は制御点を利用して裏ポリゴンモデルを変形する過程を示した図である。また、図中の赤い点は特徴点であり、青い点は制御点である。



(a) 輪郭線を描画　(b) 制御点を配置　(c) 制御点の方向に変形

図 3.5: 制御点を利用した裏ポリゴンモデルの変形

まず、2.3 節の手法を用い、裏ポリゴンモデルを利用した輪郭線を描画する。対象のモデルの裏ポリゴンモデルの輪郭線を描画した状態が図 3.5(a) である。その際に、3.1 節で検出した特徴点である頂点を裏ポリゴンモデルの頂点と互換することで、裏ポリゴンモデルで特徴点を扱えるようにする。以降、特徴点は裏ポリゴンモデルにある特徴点のことを指し、対象となるモデルの特徴点は考慮しない。

次に、輪郭線となる裏ポリゴンモデルに対して、変形する方向に制御点を配置する。図 3.5(b) は、対象のモデル付近に制御点を配置した図である。制御点は任意の位置に任意の個数配置することが可能であり、この制御点に対して特徴点の変形を行う。

さらに、すべての特徴点に対して、最も近い制御点それぞれ求める。頂点 (ポリゴン) 数自体が少なく特徴点が少ない場合や、配置する制御点が少ない場合は、特徴点と制御点を互いに関連付けることは容易である。しかし、頂点数が多く特徴点が多い場合や、制御点が多い場合は、特徴点と制御点の関連付けが煩雑になる。このため、本手法では特徴点から最も近い制御点に対して関連付けることとした。この特徴点と制御点の関連付けを保存し、次の手順の変形に利用する。

最後に、特徴点と制御点の関連付けから、頂点の法線方向に特徴点を動かすことで輪郭線の誇張表現を行う。図 3.5(c) は、特徴点である頂点を関連付けた制御点の方に移動することでモデルの変形を行った図である。このモデルの変形にあたり、特徴点の頂点座標を $\mathbf{P}$、制御点の座標を $\mathbf{Q}$ としたとき、特徴点の移動距離 $\mathbf{D}$ は計算式 (3.5) とした。また、$\mathbf{V}$ は制御点 $\mathbf{Q}$ の位置ベクトルである。

$$\mathbf{D} = \frac{\mathbf{V}}{(\mathbf{Q} - \mathbf{P})^2} \tag{3.5}$$

式 (3.5) で計算した $\mathbf{D}$ の距離だけ、特徴点を法線方向に移動することで変形を行う。また、式 (3.5) は制御点と特徴点間の距離の 2 乗による反比例の式である。このため、特徴点は制御点との距離が近いほどより大きく制御点の方に移動し、距離が遠くなるほど特徴点の移動距離は短くなる。これにより、制御点に近い特徴点部分を強調し、徐々に強調の度合いが小さくなるような滑らかな表現を行う。

3.3 実装

本手法を検証するために用いるプログラムの実装には、グラフィックス API の OpenGL[26] をベースとした 3DCG ツールキットの Fine Kernel ToolKit[27] を使用した。また、本手法を用いたプログラムの手順を次の図 3.6 に示す。

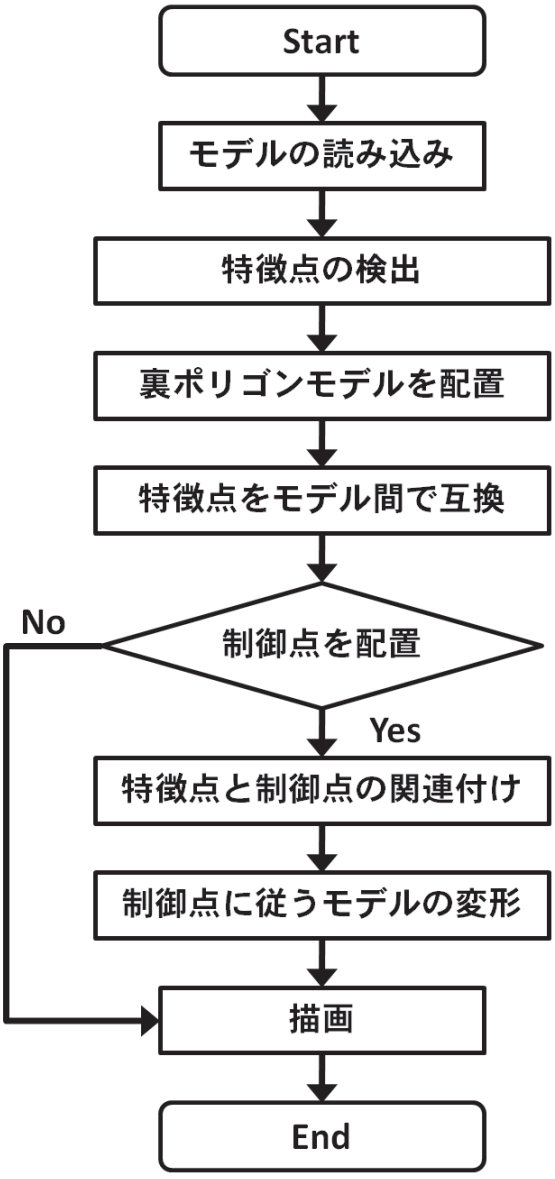


図 3.6: 処理のながれ

まず、3.1 節で述べた手法により、対象となるモデルのポリゴンの形状から特徴点を検出する。次に、元のモデルを複製し拡大することで、輪郭となる裏ポリゴンモデルを重ねて配置する。配置した裏ポリゴンモデルの頂点と元のモデルの特徴点となる頂点を互換することで、裏ポリゴンモデルで特徴点を扱えるようにする。さらに、モデルの周囲に制御点を配置している場合、3.1 節で述べた手法を用いる。特徴点から最も近い制御点を割り出し、制御点と特徴点を関連付け、最後に特徴点と制御点の距離に応じて裏ポリゴンモデルの特徴点を変形する。以上の手順で、輪郭線の誇張表現を行う。

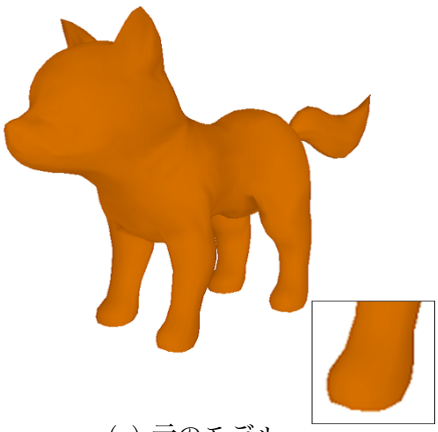
第 4 章

提案手法の検証

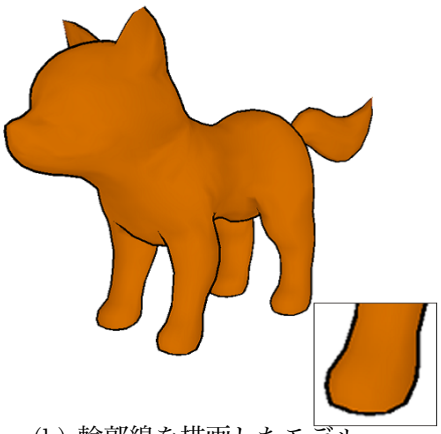
本章では、第 3 章で提案した手法を用いた結果とその検証を行う。4.1 節では、3.3 節のプログラムを利用し、本手法を 3DCG モデルに用いた結果を示す。4.2 節では本手法を用いることでどのような差異があるのかを検証し、4.3 節でリアルタイム性の検証を行う。また、3D モデルの作成には Metasequoia[29] を使用した。

4.1 本手法を用いた結果

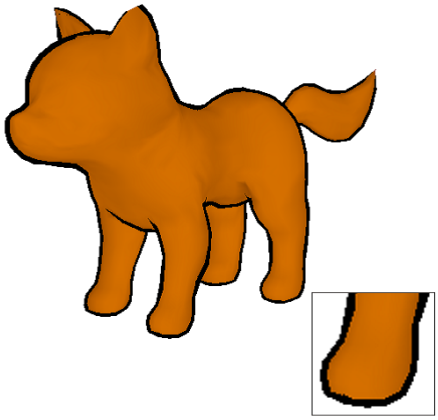
図 4.1 に、本手法を用いた結果を示す。図 4.1(a) が輪郭線を描画する前のモデル、図 4.1(b) が裏ポリゴンモデルを利用した輪郭線を描画したモデル、図 4.1(c) が本手法を用いた輪郭線を描画したモデルである。図 4.1(d) は制御点を球状のモデルとして視覚化し、図 4.1(c) に加えて表示したものである。また図 4.1(a)、図 4.1(b)、図 4.1(c) の右下の図はそれぞれの同一箇所を拡大した図である。弧を描くような箇所が制御点の方向に膨らむことで誇張している様子が確認できる。



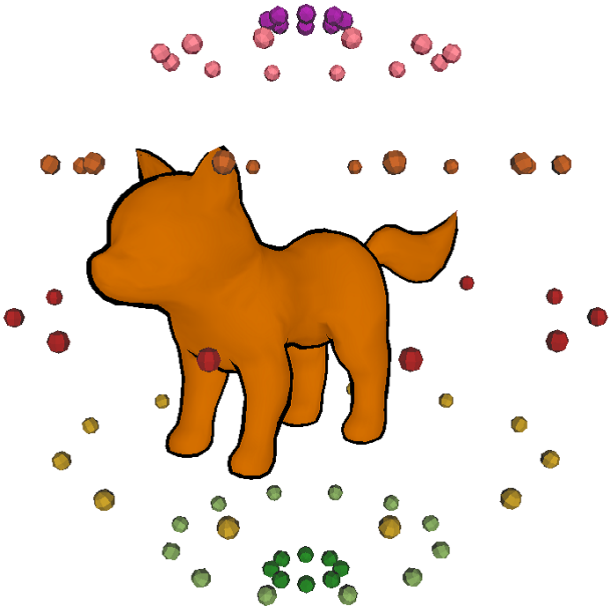
(a) 元のモデル



(b) 輪郭線を描画したモデル



(c) 本手法を適用したモデル



(d) 制御点を表示した状態

図 4.1: 本手法を用いた輪郭線の誇張表現

図 4.1 の描画結果での制御点はモデルの周囲に球状になるように配置した。配置の手順は次の通りである。なお、座標の原点は 3D モデルの中心とする。

(1) 配置する球状の半径を r とし、座標 $(r, 0, 0)$ に制御点を配置する。

(2) 座標 $(r, 0, 0)$ から z 軸を回転軸として $30°$ 回転し、制御点を配置する。この配置した位置から $30°$ 回転することを繰り返し、制御点を配置する。

(3) 座標 $(r, 0, 0)$ から y 軸を回転軸として $30°$ 回転し、(2) で行った配置を繰り返す。これを $30°$、$60°$、$-30°$、$-60°$ で行う。

(4) $80°$、$-80°$ でも (3) と同様に行うが、配置する際の回転を z 軸に対し $45°$ づつにし、制御点を配置する。

$80°$、$-80°$ にて制御点の配置を行っているのは、$90°$、$-90°$ では複数の制御点を配置できないため、ずらして配置した。

表 4.1 は、各角度において配置した制御点の個数をまとめた表である。図 4.1(d) の制御点は、表 4.1 のように配置したものである。

表 4.1: 制御点を配置した角度と個数

角度	$-80°$	$-60°$	$-30°$	$0°$	$30°$	$60°$	$80°$
制御点数	8	12	12	12	12	12	8

図 4.1(c) や図 4.1(d) で配置した制御点は、表 4.1 のように一定の条件をもとに配置した。しかし、図 4.1(d) のように特徴点の位置を条件に従って並べた場合、縦長や横長といった極端な形状の 3D モデルを使うと大きくはみ出すことがある。図 4.2 は、表 4.1 のように制御点を配置した場合に、モデルがはみ出す場合を図示したものである。高さを基準として配置し、横幅が足りずにはみ出した状態となっていることが分かる。

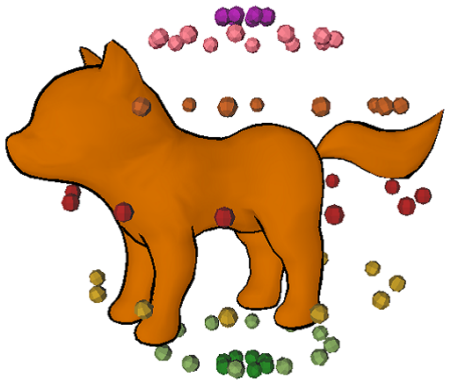
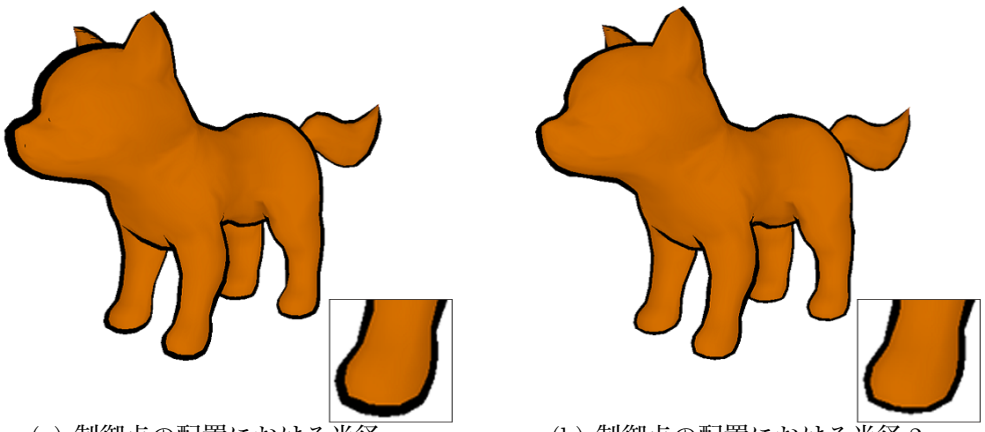


図 4.2: モデルがはみ出す例

モデルがはみ出すような制御点の配置の場合、制御点の内側にある特徴点も外側にある特徴点も変形する。しかし制御点の外側にある特徴点は、制御点と特徴点の距離が極端に近くなるなどの場合があり、予期しない変形が起きる可能性がある。このため、特徴点を意図的に変化できるよう制御点の内側にモデルがあるように制御点を配置する必要がある。また、特徴点を全て包み込むために、球状に配置する半径 r の値を変え、3D モデルを球状に大きく包み込んだ場合でも、特徴点から最も近い制御点でさえも長い距離となる可能性がある。3.2 節の式 (3.5) で示した通り、特徴点を動かす要素に特徴点と制御点間の距離があり、この距離が遠い場合特徴点の移動量は減少するため、特徴点と制御点の距離が変化すると輪郭線も変化する。

図 4.3 は特徴点と制御点間の距離による変化を比較した図である。図 4.3(a) は図 4.1(d) と同様の制御点の配置で本手法を用いた場合の図であり、図 4.3(b) は図 4.3(a) の制御点の配置に利用している半径 r を $2r$ とし、制御点を配置した場合の結果である。また、図 4.3 の輪郭線は変化をより明確とするために、式 (3.5) の裏ポリゴンモデルの法線方向に移動する距離 $\mathbf{D}$ を $2\mathbf{D}$ とした。

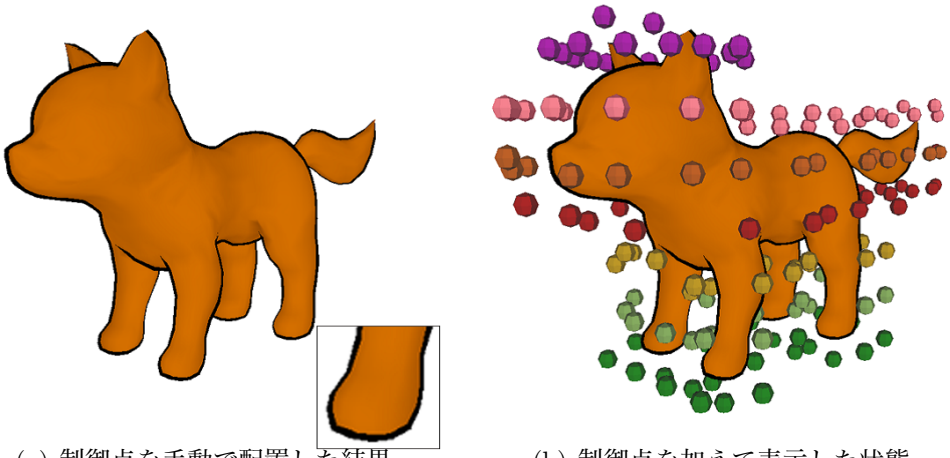


(a) 制御点の配置における半径 r　　(b) 制御点の配置における半径 $2r$

図 4.3: 特徴点と制御点間の距離による輪郭線の変化

通常の制御点の配置である図 4.3(a) の場合、裏ポリゴンモデルの変形が大きくなっていることが分かる。一方で、制御点の配置に利用する半径が $2r$ である図 4.3(b) の場合、図 4.3(a) の裏ポリゴンモデルの変形よりも、変化が小さい。このように、制御点と特徴点の距離が遠い場合、制御点に対して特徴点を動かす本手法を効果的に使用できない。このため、よりバランス良く効果的に表現する場合、モデルと制御点の距離が近すぎることがなく、また遠すぎることもない一定の距離であることが望ましい。よって、全体的にモデルと制御点の距離が一定になる配置である、モデルの形状に近い配置をすることで、より効果的な表現が可能となる。

図 4.4 は、モデルの形状に沿って手動で制御点を配置した状態を図示したものである。図 4.4(a) は形状に沿って手動で制御点を配置した場合の結果であり、図 4.4(b) は図 4.4(a) で配置した制御点を可視化した図である。



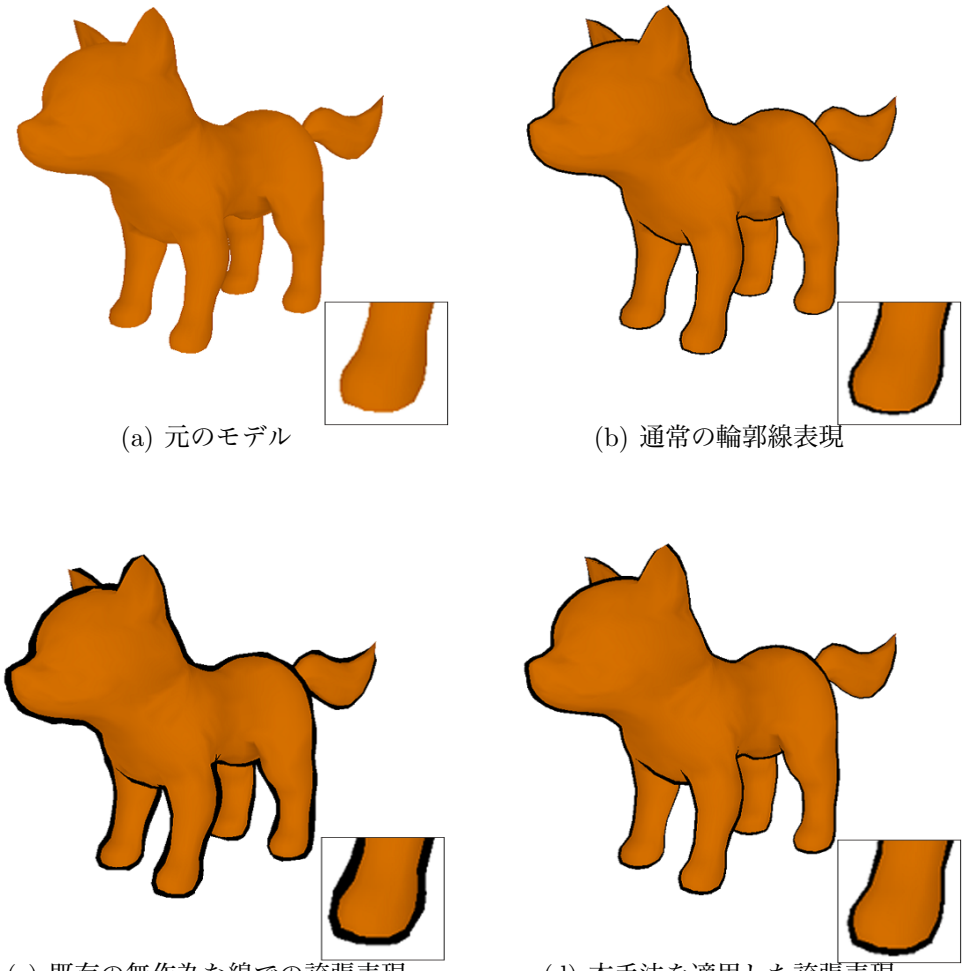
(a) 制御点を手動で配置した結果　　(b) 制御点を加えて表示した状態

図 4.4: 形状に沿って制御点を配置した場合の誇張表現

規則的に配置した図 4.1(c) は全体的に膨らんでいる様子に近いが、手動で制御点を配置した場合の図 4.4(a) は、膨らんでいる部分もあるが、通常の輪郭線の図 4.1(b) 程度の太さの線が出ている部分も確認できる。このため、効果的な表現を行っていることが確認できる。

4.2　本手法の効果を検証

本手法を用いた輪郭線の誇張表現について検証する。図 4.5 は複数の輪郭線表現を行ったモデルの画像である。図 4.5(a) は輪郭線表現をしていないモデル、図 4.5(b) が裏ポリゴンモデルを利用した輪郭線表現を行ったモデルである。図 4.5(c) が無作為に揺らすことで誇張する既存の輪郭線表現をしたモデル、図 4.5(d) が本手法を用いた輪郭線の誇張表現をしたモデルである。なお、既存の輪郭線表現は、“大神” で用いている手法 [12] を参考に、3DCG 作成ソフトウェアの Autodesk Softimage[28] にて誇張表現を行った結果を示す。また、図 4.5(a)〜図 4.5(d) の右下の図は同一部分を拡大した図である。



(a) 元のモデル　(b) 通常の輪郭線表現

(c) 既存の無作為な線での誇張表現　(d) 本手法を適用した誇張表現

図 4.5: 輪郭線の誇張表現の比較

図 4.5(b)〜図 4.5(d) を比較した結果、図 4.5(b) の通常の輪郭線表現よりも、図 4.5(d) に示す本手法の方が強弱のついた線の表現ができている。また、図 4.5(c) のような無作為に線を揺らす誇張表現では、形状を誇張するのではなく無作為に太さが変化した線で誇張している状態であり、モデルの形状を効果的に表現していない。図 4.5(d) に示すように、本手法ではモデルの形状の凹凸を誇張した輪郭線によって、より凹凸を強調するように表現している。これにより、モデルと輪郭線双方をより豊かに表現し、絵を描いた時のような自然な輪郭線の表現をしてい

ると言える。

4.3 本手法のリアルタイム性を検証

本手法のリアルタイム性を検証するために使用した環境を次の表 4.2 に示す。

表 4.2: 実行環境

OS	Windows Vista Enterprise Service Pack 2
CPU	Intel Core2 Duo E8500
メモリ	4.00GB

提案手法では、条件を満たすモデルの頂点を特徴点とし、制御点を用いて変形することで輪郭線の誇張表現を行っている。このため、頂点数の違う複数のモデルを用意し、描画速度の検証を行った。描画速度の単位は fps(Frames Per Second) であり、1 秒間当たりの描画回数を表す。表 4.3 にその結果を示す。

表 4.3: 描画速度の測定結果

頂点数	描画速度 (fps)	
	裏ポリゴンモデル	本手法のモデル
956	60.4	60.0
1790	56.0	56.0
2689	37.6	37.4
3094	27.0	27.4

表 4.3 より、頂点数が増加するにつれて描画速度が低下することがわかった。 また本手法では頂点を扱っているが、裏ポリゴンモデルを利用した手法と頂点数自体は変化しておらず、双方の手法には描画速度の差はない。このため、本研究における誇張表現手法の描画速度は頂点数に依存しているが、リアルタイム性は実現できている。よって本手法は、通常の裏ポリゴンモデルを利用する手法と描画速度に大きな差がないまま、既存手法とは異なる表現が可能であると言える。

第 5 章

考察

本章では第 4 章で行った結果と検証をもとに、本手法を用いた輪郭線の誇張表現について 5.1 節にて評価する。また、5.2 節では本手法の制約について述べ、展望を述べる。

5.1 評価

本手法を用いることによって、リアルタイム 3DCG でのトゥーンレンダリングにおける輪郭線の誇張表現が可能となった。本手法では輪郭が正確な裏ポリゴンモデルを利用した表現手法を用いた。これにより、モデルの凹凸を途切れることのない自然な線で、より手描きに近い輪郭線の表現を効果的することができた。また、3D モデルの弧を描く曲線となる部分を特徴としてモデルの形状から検出し、任意の制御点に対して変形することで輪郭線の強弱を表現した。モデルの形状によって輪郭を変化させることによって、デザインやデッサンで用いるような、線の強弱で部分的に強調し対象の物体を効果的に表現する手法を実現した。

本手法を用いることにより、ディジタル 2D アニメーション調や漫画調、イラスト調のインタラクティブなコンテンツを制作する際、より絵に近い自然な輪郭線で描画することが可能となる。また、輪郭線を意図的に誇張することが可能なため、輪郭線を太く強く描画するような劇画調などの表現としても利用することができると考えられる。

5.2 将来の展望

本研究で提案した手法には次に挙げるような制約がある。

(a) モデルのポリゴン形状によっては意図しない特徴点を検出することがある。

(b) ポリゴン数が少ない場合、輪郭線モデルが大きく変形してしまう。

(c) モデルの形状に沿って制御点を配置するのに手間がかかる。

(a) のポリゴン形状によって意図しない特徴点を検出するのは、特徴点の検出や輪郭線モデルの変形に対象となるモデルのポリゴンを利用するためである。このため、モデルのポリゴンの形状によっては特徴点の検出結果や輪郭線モデルの変形に影響が出る。また、意図しない部分を特徴点として検出することもあり、思い通りの線が出ないことがある。

(b) のポリゴン数が少ない場合に輪郭線モデルが大きく変形するのは、(a) と同様に特徴点の検出や輪郭線モデルの変形にモデルのポリゴンを利用しているためである。モデルのポリゴン (頂点) 数が少ない場合、モデル自体の凹凸が明確になりがちであり、モデルを構成するの頂点の多くが特徴点となる可能性が高い。このため、輪郭線モデルの多くの部分が変形してしまうため、意図する線が出ない場合がある。

(c) のモデル形状に沿って制御点を配置するのは、4.1 節で述べた通り、モデルの形状に沿って特徴点を配置する図 4.4(b) のように配置を行う方が、規則的に配置する図 4.1(d) のような配置より、効果的な表現が可能であるためである。しかし、形状に沿ってひとつひとつ制御点を配置するのは手間や時間がかかることである。

このように、本手法の現状ではいくつかの制約がある。これらに対し、(a) のように思い通りの線が出ない場合には、制御点を追加や消去、移動することで回避できる。しかし (a) や (b) のような、ポリゴン形状によって検出結果に影響が出たり、ポリゴン数が少ない場合、新たなモデル形状の検出手法を提案する必要があると考えられる。また、(c) のように手間がかかる作業に対しては、モデルの形状に

合わせて楕円形や円柱形のようにある程度の規則的な配置をすることで、手間を軽減できる。図 4.4(b) のようにモデルの形状に沿って配置するためには、自動的に配置する手法や、任意の位置に容易に配置可能な手法を提案する必要がある。

第 6 章

まとめ

本研究では、リアルタイム 3DCG におけるトゥーンレンダリングにて輪郭線の誇張表現を行うための手法を提案し、実現した。特に線の強弱による凹凸の表現に注目し、曲線となる部分を特徴として検出、誇張することで、対象とする物体を効果的に表現することが可能となった。また、輪郭線の描画が正確な裏ポリゴンモデルを利用する輪郭表現手法に着目することで、無作為に線を動かさず、連続した線の強弱による表現が可能となった。これにより、モデルと輪郭線の双方をより豊かに表現し、絵を描いた時のような自然な輪郭線の誇張表現を実現することが可能となった。

このため、ディジタル 2D アニメーション調や漫画調、イラスト調のインタラクティブなコンテンツを制作する際、より絵に近い自然な輪郭線で描画できる。また、輪郭線を意図的に誇張することが可能なため、輪郭線を太く強く描画するような劇画調などの表現としても利用することができる。

謝辞

本研究を締めくくるにあたり、終始温かいご指導ならびに適切な助言をくださいました、本校メディア学部 三上浩司講師と渡辺大地講師に心からの感謝の意を表します。また、研究方針の相談を引き受け、様々な助言をくださいました、片柳研究所クリエイティブラボの岡本直樹様と松島渉様、さらには研究室の院生の方々にも深く感謝致します。最後に、苦楽を共にし、歩んできた研究室のメンバーにも厚く御礼申し上げます。

参考文献

[1] Kalnins, R.D. and Markosian, L. and Meier, B.J. and Kowalski, M.A. and Lee, J.C. and Davidson, P.L. and Webb, M. and Hughes, J.F. and Finkelstein, A., “WYSIWYG NPR: Drawing strokes directly on 3D models”, ACM Transactions on Graphics, Vol.21(3), pp.755-762, 2002.

[2] Gooch, Amy and Gooch, Bruce and Shirley, Peter and Cohen, Elaine, “A non-photorealistic lighting model for automatic technical illustration”, Proc. SIGGRAPH 1998, pp.447-452, 1998.

[3] デジタルアニメ制作技術研究会, 東京工科大学, 「プロフェッショナルのためのデジタルアニメマニュアル 2009 〜工程・知識・用語〜」, デジタルアニメ制作技術研究会, pp.49-74, 2009.

[4] Guptill, A.L. and Meyer, S.E, 「鉛筆で描く」, マール社, pp.33-43, 1977.

[5] Guptill, A.L. and Meyer, S.E, 「ペンで描く」, マール社, pp.48-56, 1976.

[6] サンライズ・メ〜テレ, “バトルスピリッツ 少年突破バシン”, テレビ朝日, 2009.

[7] “JET SET RADIO”, SEGA ENTERPRISES,LTD., セガ, 2000.

[8] “ドラゴンクエスト VIII 空と海と大地と呪われし姫君”, ARMOR PROJECT, BIRD STUDIO, LEVEL-5, SQUARE ENIX, スクウェア・エニックス, 2004.

[9] Kalnins, R.D. and Davidson, P.L. and Markosian, L. and Finkelstein, A., “Coherent stylized silhouettes”, ACM Transactions on Graphics, Vol.22(3), pp.856-861, 2003.

[10] Praun, E. and Hoppe, H. and Webb, M. and Finkelstein, A., “Real-time hatching”, Proc. SIGGRAPH 2001, pp.581-586, 2001.

[11] “大神”, CAPCOM CO., LTD., カプコン, 2006.

[12] Works Corporation, 「アミューズメント映像探検隊 69 大神」, CG-WORLD(2006 年 6 月号), Vol.94, pp.85-89, 2006.

[13] Akenine-Moller, T. and Moller, T. and Haines, E., 「リアルタイムレンダリング 第 2 版」, ボーンデジタル, pp.247-267, 2006.

[14] Gooch, B. and Sloan, P.P.J. and Gooch, A. and Shirley, P. and Riesenfeld, R., “Interactive technical illustration”, Proceedings of the 1999 symposium on Interactive 3D graphics, pp.31-38, 1999.

[15] Mark DeLour, 「Game Programing Gems 2 日本語版」, ボーンデジタル, pp.436-443, 2002.

[16] Saito, T. and Takahashi, T., “Comprehensible Rendering of 3-D Shapes”, Computer Graphics, pp.197-206, 1990.

[17] Decaudin, P., “Cartoon-looking rendering of 3D-scenes”, 1996.

[18] Hyunjun Lee and Sungtae Kwon and Seungyong Lee, “Real-time pencil rendering”, NPAR 2006, pp.37-45, 2006.

[19] Thomas Luft and Oliver Deussen, “Real-time watercolor illustrations of plants using a blurred depth test”, NPAR 2006, pp.11-20, 2006.

[20] Card, D. and Mitchell, J.L., “Non-photorealistic rendering with pixel and vertex shaders”, ShaderX: Vertex and Pixel Shaders Tips and Tricks, pp.319-333, 2002.

[21] Mitchell, Jason L. and Brennan, Chris and Card, Drew, “Real-time image-space outlining for non-photorealistic rendering”, Proc. SIGGRAPH 2002, pp.239-239, 2002.

[22] Rossignac, J. and van Emmerik, M., “Hidden contours on a frame-buffer”, 1992.

[23] Evan Hart, Dave Gosselin, John Isidoro, “Vertex Shading with Direct3D and OpenGL”, 2001.

[24] 鈴木隼人, “米国マンガ調セルシェーディングに関する研究”, 東京工科大学大学院 メディア学研究科 修士論文, 2004.

[25] 地神知哉, “3DCG におけるキャラクターの表示サイズによる漫画的簡略化表現手法”, 東京工科大学 メディア学部 卒業論文, 2006.

[26] Silicon Graphics International., “OpenGL”, <http://www.opengl.org/>.

[27] 渡辺大地, “Fine Kernel Tool Kit System”, <http://fktoolkit.sourceforge.jp/>.

[28] Autodesk, Inc., “Autodesk Softimage”, <http://www.autodesk.co.jp/>.

[29] Osamu Mizuno, “Metasequoia”, <http://www.metaseq.net/>.