修士論文　　平成 23 年度 (2011)

リアルタイム 3DCG における物体の形状を考慮した輪郭線の誇張表現手法の提案

東 京 工 科 大 学 大 学 院
バイオ・情報メディア研究科 メディアサイエンス専攻

松尾 隆志

修士論文　　平成 23 年度 (2011)

リアルタイム 3DCG における物体の形状を考慮した輪郭線の誇張表現手法の提案

指導教員　三上 浩司　専任講師

東 京 工 科 大 学 大 学 院
バイオ・情報メディア研究科 メディアサイエンス専攻

松尾 隆志

論 文 の 要 旨

論文題目	リアルタイム 3DCG における物体の形状を考慮した 輪郭線の誇張表現手法の提案
執筆者氏名	松尾 隆志
指導教員	三上 浩司　専任講師
キーワード	3DCG，トゥーンレンダリング，輪郭線，誇張表現

[**要旨**]

2D アニメーションや漫画では，物体の形状をより効果的に表現するために，輪郭線の線質を変えるなどの誇張した表現で描くことがある．3DCG 上で 2D アニメーションや漫画の表現を行うトゥーンレンダリングでは，輪郭線は容易な手法である均一な線で表現することが多い．本研究は，ゲームなどのリアルタイムコンテンツ内でのトゥーンレンダリングにおける，物体の形状を考慮した輪郭線の誇張表現手法を提案する．本手法では，2D アニメーションの作画に用いるデザインやデッサンの手法である，輪郭線の強弱の変化による形状の誇張表現に着目した．

1 つ目の提案手法では，対象となる 3 次元モデルのポリゴン形状が曲線を描く部分を特徴とし，その特徴に従って任意に操作できる点を配置することで誇張表現を行った．また，3 次元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ることで，形状に沿う連続した輪郭線の表現を行った．これにより，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することができた．しかし，3 次元モデルの特徴検出の不正確さや，3 次元モデル自体の形状変形には対応し難い問題点が残った．

2 つ目の提案手法では，対象となる 3 次元モデルのポリゴン形状の曲率を計算することで特徴を検出し，輪郭線の太さを曲率に沿って変化することで誇張表現を行った．また，この手法でも裏ポリゴンモデルを利用することにより，形状に沿う連続した輪郭線の表現を行った．これにより，この手法においてもリアルタイム 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することができた．特徴検出をより正確に行い，かつ形状変形にも対応した輪郭線の誇張表現が可能となった．

これらことより，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した．

A b s t r a c t

Title	Shape Oriented Line Drawing in 3DCG
Author	Takashi Matsuo
Advisor	Assistant Professor Koji Mikami
Key Words	3DCG, Toon-Rendering, Outline, Exaggreration

[**summary**]

In 2D animation and comics, the line quality of object outlines and contour lines are often varied to emphasize or exaggerate the shape of an object. One such technique is to vary the line thickness depending on the object shape. This technique is often used in sketching and drawing to give objects a greater sense of depth or dimension. When using toon rendering to simulate a 2D-esque effect in 3D animation, techniques that render object outlines of constant thickness are popular due to their ease of application. While existing methods allow for a certain level of dynamic line thickness through the use of shading, they require long calculation times and are not suited for real time rendering. Our study aims to provide a viable real-time 3D toon rendering technique that creates object outlines with varying thickness.

In the first method, we focused on exaggerating thickness of lines where the target polygon is curved. Furthermore, by using an backface polygon of the target polygon, we are able to create a continuous outline of the object, thus recreating a closed outline of object with varying line thickness. In doing so, we accomplished our goal of developing a toon rendering method that successfully achieves a varying object outline thickness within a real-time 3DCG environment. However, the problem is incorrect and feature detection in 3D models and can not accommodate itself to the shape deformation of 3D models.

In the second method, points of curvature on a 3D model correspond to places where the model has either a concave or convex surface. We calculate the mean curvature of each vertex point on the 3D model, enabling us to identify the shape feature of any given vertex on the model. In addition, this method involves creating a slightly enlarged copy if the original 3D model with reversed normals. Object outline thickness is changed by moving the vertices on the copied model outward (away from the original model). Vertices with a high amount of curvature are moved more than those with a low amount of curvature. The result is a dynamic and smooth object outline with varying thickness. This was made possible by calculating the curvature of the 3D model, creating an enlarged copy of the model with reversed normals, and manipulating the copy based on the curvature.

From these it was realized that the outline exaggeration effectively represented in the form of real-time toon rendering 3DCG.

目 次

図 目 次

表 目 次

第 1 章

はじめに

1.1 研究背景

近年，鉛筆画や水彩画などの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) [1][2] の研究が盛んに行われている [3][4][5][6][7]．その中に，漫画や 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 (セルレンダリング)[8][9] という手法がある．トゥーンレンダリングは，数階調の陰影からなるシェーディングと，細い輪郭線で表現する手法である．特に日本では手描きアニメーションと 3DCG を組み合わせる際に，互いを調和する目的でトゥーンレンダリングを用いている [10]．次の図 1.1 にトゥーンレンダリングの例を示す．

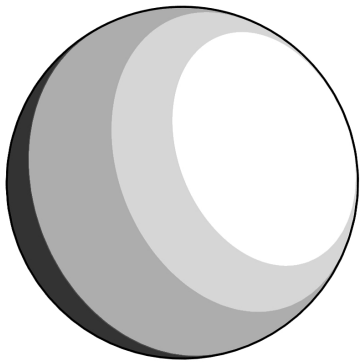


図 1.1: トゥーンレンダリングの例

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着目した．

本来，人は物体を見るとき，明度や彩度，色彩によって形を認識している．これに加え，元々物体にはない輪郭線を描くことで，より簡単に認識でき，意図的に形状や差を強調することができる．この輪郭線は漫画や 2D アニメーションに利用されている．特にディジタル 2D アニメーションでは，均一な太さの輪郭線が多く用いられている．図 1.2 にディジタル 2D アニメーションにおける均一な太さの輪郭線を用いた例を示す．

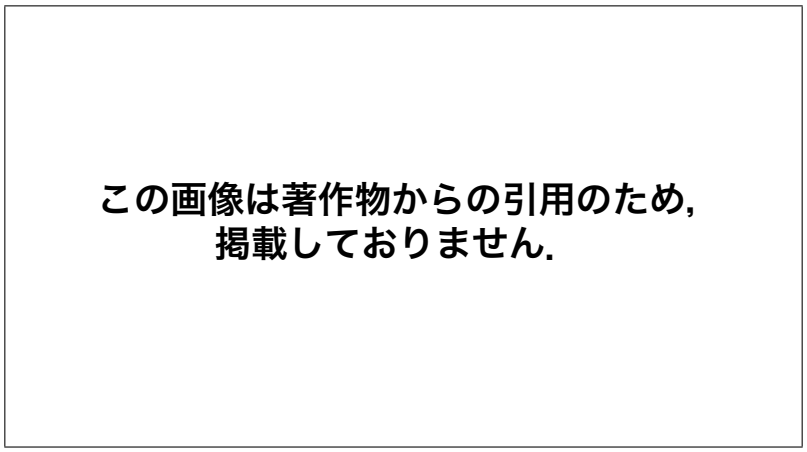


図 1.2: ディジタル 2D アニメーションにおける均一な太さの輪郭線を用いた例
出典: “サマーウォーズ”

これは，紙に作画した線をコンピュータ上にスキャンするときに，コンピュータ上で太さが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も，処理が最も単純である太さが均一な線を多く用いている．

しかし，実際の漫画や 2D アニメーション作品では，イラストや水彩などの様々な線の表現を再現するために，線の太さの強弱や色の濃淡などで輪郭を誇張表現することがある．次の図 1.3 に，実際に輪郭線を誇張表現している 2D アニメーションの例を示す．

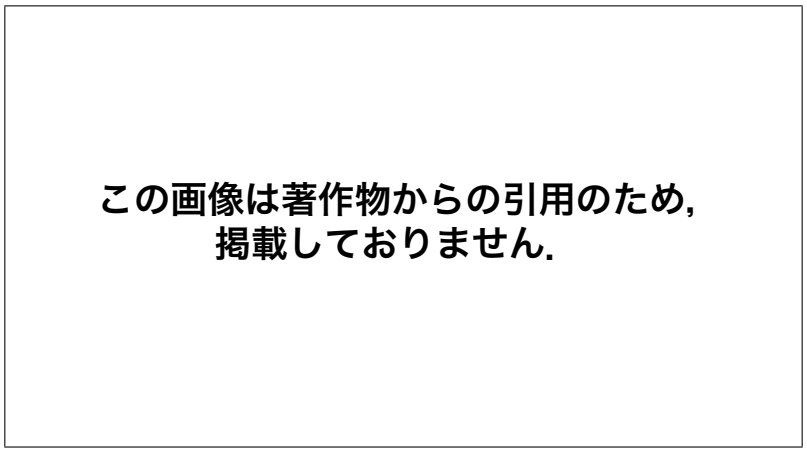


図 1.3: ディジタル 2D アニメーションで輪郭線の誇張表現を用いた例
出典: “映画ドラえもん 新・のび太と鉄人兵団 〜はばたけ 天使たち〜”
ⓒ藤子プロ・小学館・テレビ朝日・シンエイ・ADK 2011

この誇張する表現によって，輪郭や形状をより多様に表現することが可能となる．ただし，こうした輪郭線表現を 2D アニメーション上で行うためには，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必要がある．3DCG 上においても同様に，輪郭線の太さが変化するような処理や描画 [11][12][13] を行う必要がある．このような処理のためにプリレンダリングという手法を用いているが，計算や処理に時間がかかるという課題がある．

一方，リアルタイムレンダリングは，1 秒間に 30〜60 フレームの描画速度を維持するという制約の中で表現を行う手法である．リアルタイムレンダリングでは高速に処理する必要があり，プリレンダリングと同様の手法や表現を行うことは難しい．これより，リアルタイムレンダリングでのトゥーンレンダリングの多くは，描画処理に影響が少なく容易な手法である，均一な線で表現することが多い．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均一な太さの輪郭線の例を次の図 1.4 に示す．

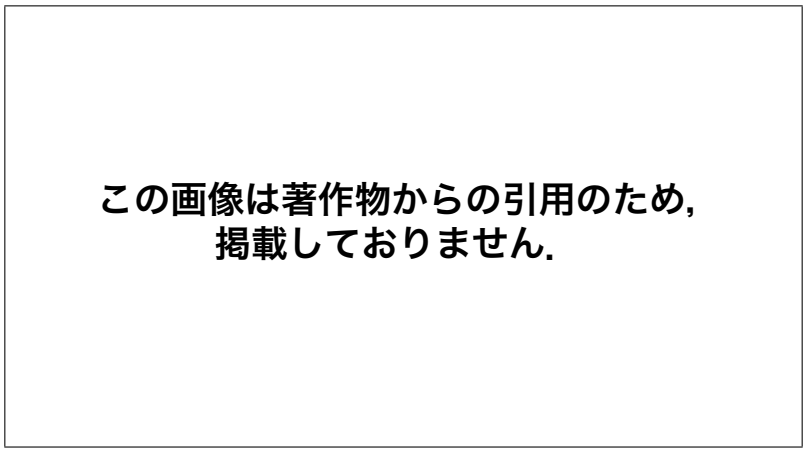


図 1.4: リアルタイムレンダリングでのトゥーンレンダリングの例
出典: “二ノ国 白き聖灰の女王” ©2011 LEVEL-5 inc.

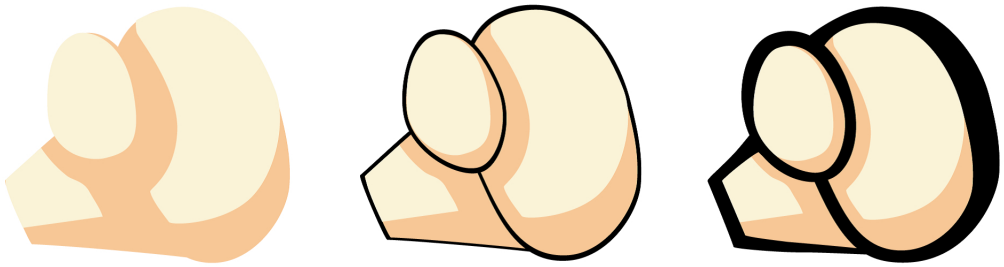
このことより，プリレンダリングで行っている高品質な太さの変化がある輪郭線表現が，リアルタイムレンダリング上で求められている．

この課題を解決するために本研究では，リアルタイム 3DCG でのトゥーンレンダリングにおいて，物体の形状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．本研究では輪郭線の誇張表現をリアルタイム 3DCG 上で実現するために，次の 3 項目を目的として設定した．

- リアルタイム 3DCG でのトゥーンレンダリングにおいて輪郭線の表現を行うこと
- 輪郭線の太さの変化によって 3D モデルの形状をより効果的に表現すること
- 連続した太さの強弱の変化によってより手描きに近い輪郭線の表現を行うこと

また，本研究では輪郭線の誇張表現に関し，線の強弱の表現に注目した．線の強弱の表現は，2D アニメーションにおいて作画する際に用いる，デッサンやデザインによる手法である．線の強弱による表現は，線の太さを変えるというシンプ

ルな手法であり，描き手のストロークや筆圧が直接線として出る手法である [14]．また，曲線や奥行きを表現する際に用い，丸みや立体感，強調といった形状を誇張する効果を与える [15]．この線の強弱による表現は，形状を効果的に表現する手描きならではの手法である．次の図 1.5 に輪郭線を比較したイラストを示す．



(a) 輪郭線なし　(b) 均一な太さの輪郭線　(c) 線の強弱による誇張表現

図 1.5: 輪郭線の比較

図 1.5(a) は輪郭線の無いイラスト，図 1.5(b) が均一な太さの輪郭線を用いたイラスト，図 1.5(c) が線の強弱による誇張表現を用いたイラストである．図 1.5(b) と図 1.5(c) を比較すると，図 1.5(c) の方がより丸みを帯びた表現となっている．

本研究ではこの線の強弱に関し，3 次元形状上で曲線となる部分を対象とし，連続した太さの変化のある輪郭線をリアルタイム 3DCG 上で行うこととした．提案手法では，3 次元形状上で曲線となる部分を特徴とし，3 次元形状から特徴の検出を行った．また，ポリゴンの裏側を用いた輪郭線描画手法に検出した特徴を反映することで，リアルタイムレンダリング上で輪郭線の太さ変化表現を実現した．さらに，提案手法を実装したプログラムを用いて評価実験を行い，提案手法の有用性を評価した．実験結果から，設定した 3 項目の目的を達成し，リアルタイム 3DCG 上で輪郭線の誇張表現を行うことを確認した．

1.2 論文構成

本研究では，リアルタイム 3DCG における形状を考慮した輪郭線の誇張表現手法を実現することを目的とし，2 つの輪郭線の誇張表現手法について研究を行った．

第 2 章では，本研究における従来手法及び研究対象について述べる．第 3 章では，3 次元形状の特徴を示す特徴点と誇張する方向を制御する制御点を用いた輪郭線誇張表現手法について述べる．第 4 章では，3 次元形状の曲率を利用した輪郭線誇張表現手法について述べる．そして第 5 章では，本研究を通じた研究の成果と今後の展望について述べる．

第 2 章

従来手法と研究対象

2.1 従来手法

本研究では，2D アニメーションで用いる，輪郭線の太さの変化する誇張表現に着目した．トゥーンレンダリングで用いるような 3 次元形状 (以下「3D モデル」) を利用する輪郭線の表現について次の 3 点に分けて述べる．

(1) 3D モデルを用いた均一な太さの輪郭線の表現手法

(2) 3D モデルを用いた太さ変化のある輪郭線の表現手法

(3) NPR における絵画風の輪郭線の表現手法

この 3 項目に対し，均一な太さの輪郭線表現の従来手法について 2.1.1 節で述べ，太さ変化のある輪郭線表現の従来手法について 2.1.2 節で述べ，絵画風の輪郭線表現の従来手法について 2.1.2 節で述べる．特に，従来手法の特徴と本研究で取り扱う課題について述べる．

2.1.1 3D モデルを用いた均一な太さの輪郭線の表現手法

3D モデルに対して太さが均一な輪郭線を描画する手法に関する研究は多くある．

Saito ら [16] は，奥行きの情報を保持する深度 (デプス) バッファと，ポリゴンの法線情報を保持する法線 (ノーマル) バッファを利用した，輪郭線の描画手法を提案した．Saito らの手法では，まず深度バッファと法線バッファをそれぞれテクスチャとして保存し，テクスチャに対して 1 次微分フィルタである Sobel フィルタの処理を行うことで輪郭線をそれぞれ抽出する．この抽出したそれぞれの輪郭線画像を合成し，画面に乗算することで，均一な太さの輪郭線として描画した．Decaudin[8] は，この Saito らの手法を改良し，トゥーンレンダリングの輪郭線の描画手法として用いた．次の図 2.1 に Saito らの手法での輪郭線の例，図 2.2 に Decaudin の手法での輪郭線の例を示す．

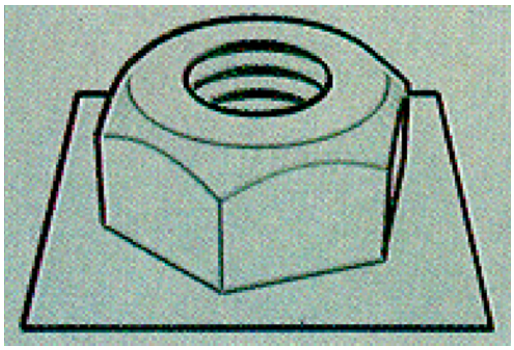


図 2.1: Saito らの手法 [16] での輪郭線

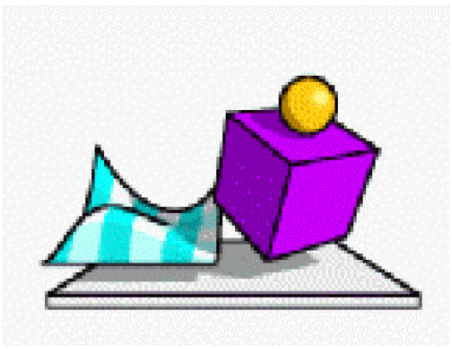
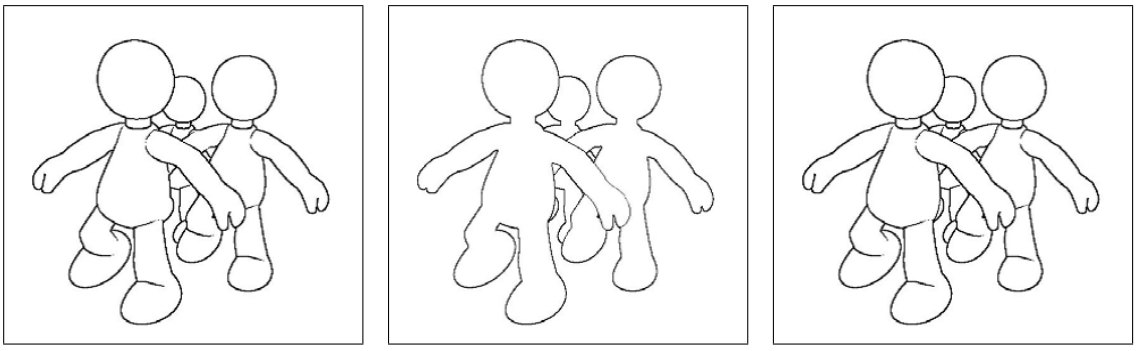


図 2.2: Decaudin の手法 [17] での輪郭線

Card ら [17] や Mitchell ら [18] は，Saito らの手法をグラフィックスハードウェアを用いた手法に改良し，リアルタイムレンダリングでの手法を実現した．図 2.3 に Mitchell らの手法での輪郭線の例を示す．図 2.3(a) が深度バッファから得た輪郭線で，図 2.3(b) が法線バッファから得た輪郭線であり，図 2.3(c) が図 2.3(a) と図 2.3(b) を合成した輪郭線の例である．

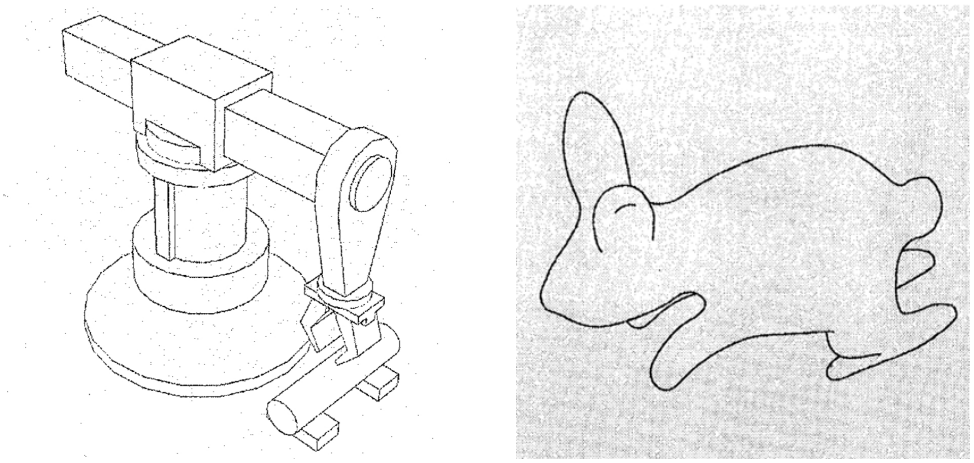


(a) 深度バッファからの輪郭線　(b) 法線バッファからの輪郭線　(c) (a) と (b) を合成した輪郭線

図 2.3: Mitchell らの手法 [18] での輪郭線

また，望月ら [19][20] は，輪郭を特徴ごとに分類し，特徴ごとに太さや色の情報を指定する輪郭線の描画手法を提案した．望月らの手法では輪郭の特徴として，輪郭を形成するポリゴン面が 1 つだけ可視状態の輪郭を輪郭線，2 つとも可視状態の輪郭を内形線と定義した．また，輪郭線と内形線の関係から，輪郭線は内形線よ

り太く，色が濃いという規則を設定した．これらの特徴や規則を基に，描画した画像に対して処理を行うことで，ポリゴンで構成した 3D モデルや，ポリゴンを分割することで生成する細分割曲面において，輪郭線の描画を実現した．次の図 2.4 に望月らの手法による輪郭線の例を示す．図 2.4(a) はポリゴンメッシュに手法を適用した例，図 2.4(b) は再分割曲面に手法を適用した例である．



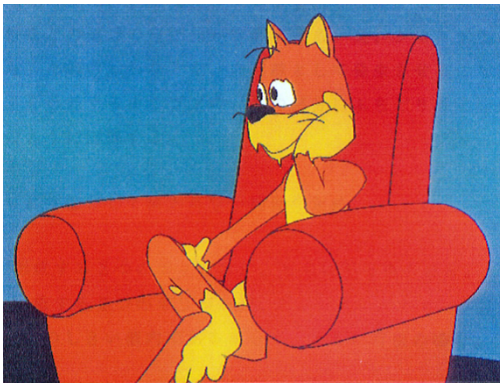
(a) ポリゴンメッシュ [19] での輪郭線　　(b) 再分割曲面 [20] での輪郭線

図 2.4: 望月らの手法 [19][20] での輪郭線

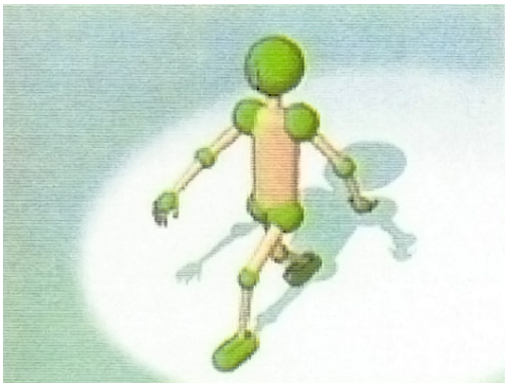
金子 [21] は，3DCG を利用した 2D アニメーション制作手法を提案し，その中でシルエット法，エッジ検出法，Z ライン法の 3 種類の輪郭線表現手法を提案した．まずシルエット法は，対象となる 3D モデルを描画した画像のモデル部分のピクセルから領域を生成し，その領域を黒く塗りつぶした上で少しだけ拡大する．その後，対象となる 3D モデルを描画した結果と合成することで輪郭線を描画する手法である．次にエッジ検出法は，3D モデルをレンダリングした画像に対し，2 次微分フィルタである Laplacian フィルタを用い輪郭を検出し，輪郭線の描画を行う手法である．最後に Z ライン法は，望月ら [22] が提案した手法を拡張したもので，隣接する 2 つのポリゴンの可視状態とポリゴンの色によって境界を定義し，境界となる箇所に輪郭線を描画する手法である．次の図 2.5 に，金子らの手法を用いた輪郭線の例を示す．図 2.5(a) がシルエット法，図 2.5(b) がエッジ検出法，図 2.5(c)

がZラインで輪郭線を描画した例である．

(a) シルエット法の輪郭線



(b) エッジ検出法の輪郭線



(c) Z ライン法の輪郭線

図 2.5: 金子の手法 [20] での輪郭線

Rossignac ら [23] は，輪郭線がポリゴン面の前になるよう深度情報を用いた輪郭線描画手法を提案した．Rossignac らの手法では，深度情報を用いることでワイヤーフレーム状の輪郭線や隠線を描画した．また，Raskar ら [24] は，Rossignac らの手法を基に，3D モデルの背面を利用した輪郭線表現手法を提案した．Raskar らの手法は，3D モデルの前面を描画した後，背面となる面を拡大し，黒く塗りつぶすことで輪郭線とする手法である．後に，Raskar[25] はグラフィックスハードウェアを利用することでリアルタイムに輪郭線を描画する手法も提案した．次の図 2.6 に Rossignac ら [22] の手法を用いた輪郭線の例，図 2.7 に Raskar ら [24] の手法を用いた輪郭線の例，図 2.8 に Raskar[25] の手法を用いた輪郭線の例を示す．

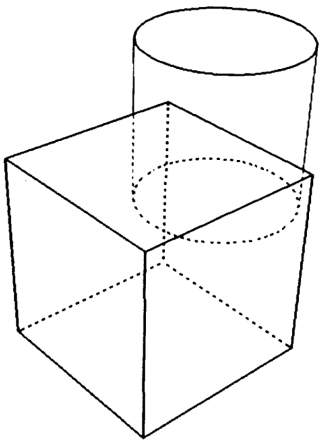


図 2.6: Rossignac らの手法 [23]

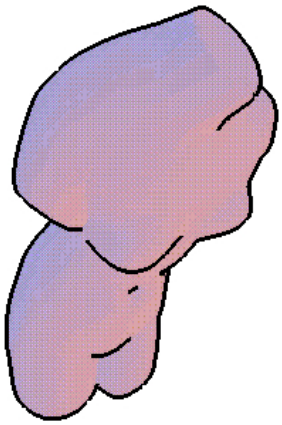


図 2.7: Raskar らの手法 [24]

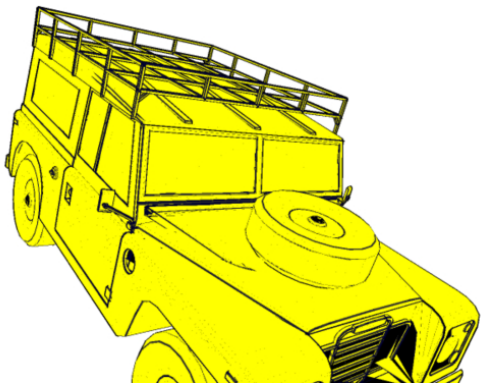


図 2.8: Raskar の手法 [25]

2.1.2 3D モデルを用いた太さ変化のある輪郭線の表現手法

太さの変化のある輪郭線表現を 2D アニメーション上で行うためには，手作業によるレタッチなどの手間を加える必要がある．3DCG 上でも同様に，輪郭線の太さが変化するような処理や描画の手間を加える必要がある．

Gooch ら [26] は，ポリゴン面に垂直なベクトルである法線ベクトルと視点の方向ベクトルである視線ベクトルの内積の値を利用する輪郭線の描画手法を提案した．あるポリゴン面の法線ベクトルに対し，視線ベクトルの内積の値が 0 に近い場合，このポリゴンは視点から見て真横に近いと判断できる．このため，内積の

値が 0 に近い場合に輪郭線とみなし，ポリゴンを黒く塗ることによって輪郭線の描画を実現した．Marshall[27] は，Gooch らの手法をグラフィックハードウェア上で行い，リアルタイムで実行する手法を提案した．次の図 2.9 に，Gooch らの手法を用いた輪郭線の例を示す．

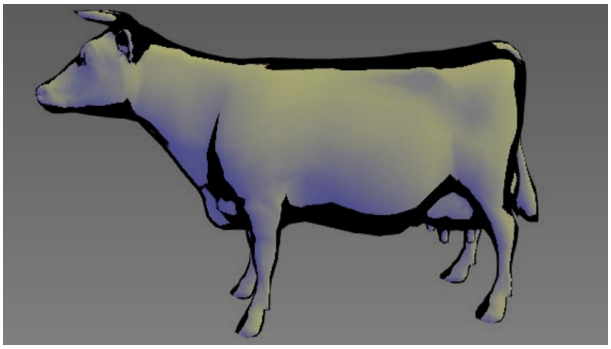


図 2.9: Gooch らの手法 [26] での輪郭線

Goodwin ら [28] は，頂点の法線ベクトルと頂点から視点方向へのベクトルを利用した太さ変化のある輪郭線描画手法を提案した．ある頂点に隣接するポリゴン面の法線を平均したものを頂点の法線ベクトルとし，この頂点の法線ベクトルとある頂点からの視点方向へのベクトルの内積を計算する．この内積の値が 0 から任意の値となる頂点を輪郭とみなし，黒く塗りつぶすことで太さ変化のある輪郭線を描画を実現した．またこれに加え，Goodwin らは内積の値で求める式に，対象の頂点における深度値や曲率半径，焦点距離を反映することで，奥行きによる太さの変化も実現した．次の図 2.12 に，Goodwin らの手法を用いた輪郭線の例を示す．

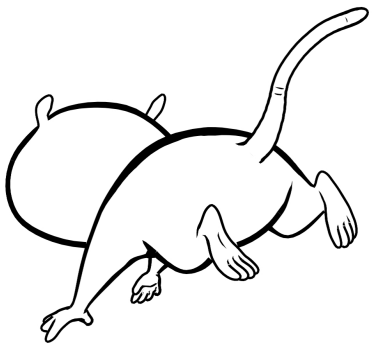


図 2.10: Goodwin らの手法 [28] での輪郭線

また,「大神 [29]」のように処理を加えることで，太さ変化のある輪郭線の表現を行っているゲーム作品の例もある．次の図 2.11 に,「大神」における誇張した輪郭線表現の例を示す.

この画像は著作物からの引用のため,
掲載しておりません.

図 2.11: ゲーム作品における誇張した輪郭線の表現の例
出典: “大神” ©CAPCOM CO., LTD. 2006, 2008.

この作品での手法は，Raskar ら [24] の手法で描画した輪郭線に，無作為な変化を加える処理 [30] を加えている．これにより，図 2.11 のような輪郭線の太さの変化を実現している.

2.1.3　NPR における絵画風の輪郭線の表現手法

手で描いたような質感の画像を再現する NPR の分野での絵画調の表現をする研究においても，均一な太さの輪郭線や輪郭線を誇張するような表現を実現している．

村上ら [31] は，パステル画の表現を 3DCG 上で行う手法を提案し，その中でパステル画調の輪郭線の描画手法を実現した．村上らの手法ではまず，Gooch らの手法 [28] を利用することで輪郭を検出する．この輪郭に対し，実際のパステル画の画材や特徴から定義した，パステル画調のストロークモデルを適用し，輪郭線を描画する．また，実際のパステル画における筆圧による減衰や，顔料の付着率も考慮することで，輪郭線の減衰やかすれの表現を実現した．次の図 2.12 に，村上らによる 3DCG 上でのパステル画表現の例を示す．

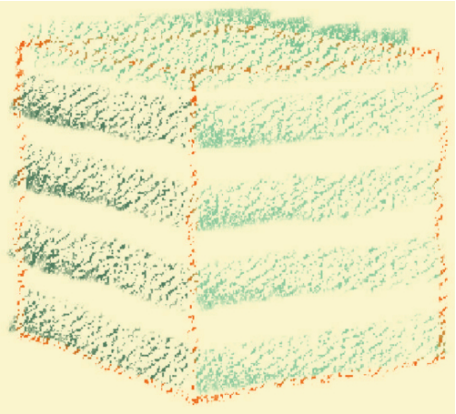


図 2.12: 村上らの手法 [31] での輪郭線

このような村上らの手法のように，絵画調の手法を 3DCG 上で再現する際に輪郭線の誇張表現を行う研究は多くある [32][33]．

Northrup ら [34] は，一度描画した画像に対して，テクスチャをマッピングすることでの輪郭線表現手法を提案した．まず，Gooch ら [26] の手法を用いて輪郭を検出し，輪郭部分の色だけを変えて描画する．次に，描画した画像から輪郭線の重なりを結合し，繋がった線として形成し，輪郭として扱うこととする．最後に，

形成した輪郭に沿ってイラストタッチの線が描かれた輪郭線用のテクスチャを貼り付けることで，イラストタッチの輪郭線表現を実現した．この輪郭線用のテクスチャを変えることにより，太さが一定な輪郭線や太さ変化のある輪郭線表現が自由に可能である．次の図 2.13 に，Northrup らによる輪郭線表現の例を示す．

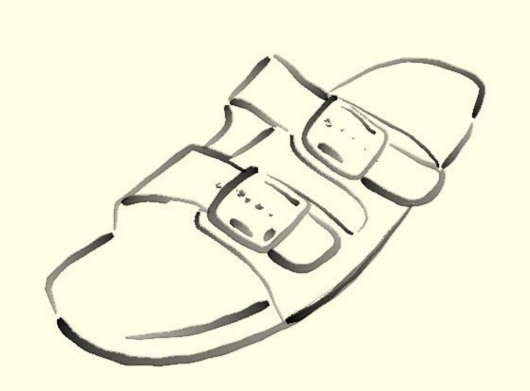


図 2.13: Northrup らの手法 [34] での輪郭線

このような Northup らの手法のように，輪郭線用のテクスチャを用意することで輪郭線の誇張表現を行う研究は多くある [35][36]．

また，Kalnins らは [37][38]，画面上に描いた絵や線を直接 3D モデルに反映する手法を提案し，その中で画面上に描いた線を輪郭線として描画する手法を提案した．Northrap らの手法 [34] をベースとし，画面上に描いた線をテクスチャとして 3D モデルの輪郭に貼り付けることで，ユーザの入力による輪郭線表現を実現した．

次の図 2.14 に，Kalnins らによる輪郭線表現の例を示す．

図 2.14: Kalnins らの手法 [37] での輪郭線

2.2 研究対象

2.1 節で述べた輪郭線の表現手法の特徴を次の表 2.1 にまとめる．

表 2.1: 輪郭線の表現手法ごとの特徴

手法	連続した線	太さ変化	連続した太さ変化	
			プリレンダリング	リアルタイム
均一な輪郭線	◯	×	×	×
変化のある輪郭線	◯	◯	◯	×
絵画風の輪郭線	×	◯	◯	×

すべての手法において輪郭線の描画は可能であるが，それぞれの手法においての特徴に違いがある．

まず，均一な太さの輪郭線の表現手法では，Saito ら [16] や，金子 [21] や，Rossignac ら [23] の手法によって，途切れることのない連続した線で均一な太さの輪郭線を実現しており，Card ら [17] や Mitchell ら [18] らの手法では，リアルタイムでの描画も実現している．また，望月ら [19][20] の手法では，輪郭線と内形線では太さは違うものの，輪郭線ごとや内形線ごとの太さは一定であり，均一な太さの輪郭線の描画を実現している．しかし，実際の 2D アニメーションなどでは均一の輪郭線が多くあるが，様々な輪郭線の表現を再現するために，線の強弱や濃淡などの変化で輪郭を誇張表現することがある．この誇張表現を既存手法で行うためには，これらの手法とは別の手法を加えることが必要となる．

次に，3DCG 上で太さ変化のある輪郭線表現の場合では，Gooch ら [26] や Goodwin ら [28] の手法によって太さ変化のある輪郭線が実現しており，段階的に太さが変化するような連続した太さ変化も実現している．しかし，Gooch ら [26] の手法では，ポリゴン面と視線ベクトルの角度によって輪郭かどうかを判断するため，ポリゴン面の角度や視線の方向よっては面全体が輪郭線となり，黒く塗りつぶす場合がある．次の図 2.15 に面全体が輪郭線となり，黒く塗りつぶした場合の例を示す．

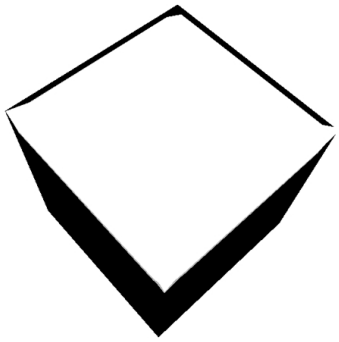


図 2.15: 面の全体が輪郭線となった例

図 2.15 のように面全体が黒くなる状態は，輪郭線とはいえない．また Goodwin ら [28] の手法では，形状に沿って連続した太さ変化を実現しているが，リアルタイムレンダリングではなくプリレンダリングで時間をかけて計算して行っているため，この手法をそのままリアルタイムレンダリングで利用することはできない．一方で,「大神」のように連続した太さ変化をリアルタイムで行っている事例もあるが，この手法は輪郭線を無作為に変化する表現であり，対象となる物体の形状を効果的には表現するものとはなっていない．このため，リアルタイムレンダリング上で輪郭線の太さの変化を用いて，3D モデルの形状をより効果的に表現する手法が望まれている．

また，絵画風の輪郭線の表現手法では，村上ら [31] の手法を始めとしたさまざまな絵画風の表現手法で，輪郭線の太さの変化を実現しており，手描きのような輪郭線の表現を実現している．しかし，絵画手法風の表現をトゥーンレンダリングにおいてそのまま用いた場合，輪郭線が絵画調や絵画手法に則すものとなり，多くの場合では連続した線ではなく途切れた線の表現となる．このため，連続した輪郭線を作画し表現する，2D アニメーション調の表現とは異なるものとなる．また，Northrup ら [34] や，Kalnins ら [37][38] の手法においても，テクスチャを利用することで連続した変化のある輪郭線を実現しているが，途切れた線の表現であるため，2D アニメーション調の表現とは異なる．このため，途切れることの無い連続

した輪郭線の強弱変化で，手描きに近い輪郭線の表現を行うことが必要である．

以上の従来手法の成果と課題より，リアルタイム 3DCG のトゥーンレンダリングでは，3D モデルの形状をより効果的に表現することや，連続した輪郭線の太さ変化を表現することが望まれている．このため本研究では，1.1 節で述べた 3 項目を満たす，連続した輪郭線の太さ変化による誇張表現手法を提案する．

本研究では，線の強弱を表現する対象として，曲線を描く部分を対象とした．3DCG 上で曲線となる部分は，3D モデルにおける凹形部分と凸形部分である．図 2.16 は 3D モデルにおける凹凸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．

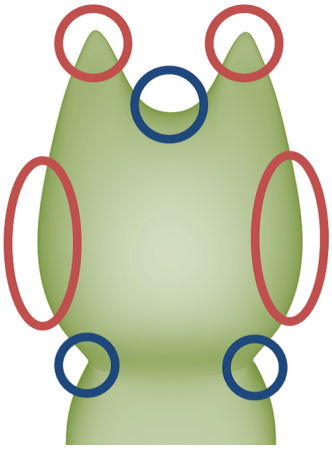


図 2.16: モデルの凹凸部分

図 2.16 の青い丸，赤い丸で示す箇所のような特徴を検出し，特徴に応じて輪郭線の変化を付けることにより，連続した変化のある輪郭線の誇張表現を行う．

本論文では，このような 3D モデルにおける凹凸に応じて輪郭線に強弱を加える表現をリアルタイム 3DCG 上で行う手順を提案する．

第 3 章

特徴点と制御点を用いた輪郭線の誇張表現手法

3.1　はじめに

本章では，特徴点と制御点を用いた輪郭線の誇張表現手法について述べる．まず，3.2 節で提案手法を述べ，3.3 節で提案した手法を実装したプログラムを用いた検証を述べる．最後に 3.4 節でまとめを述べる．

3.2　特徴点と制御点を用いた輪郭線の誇張表現手法

3.2.1　手法の概要

図 3.1 は，本章での手法の概略を示した図である．

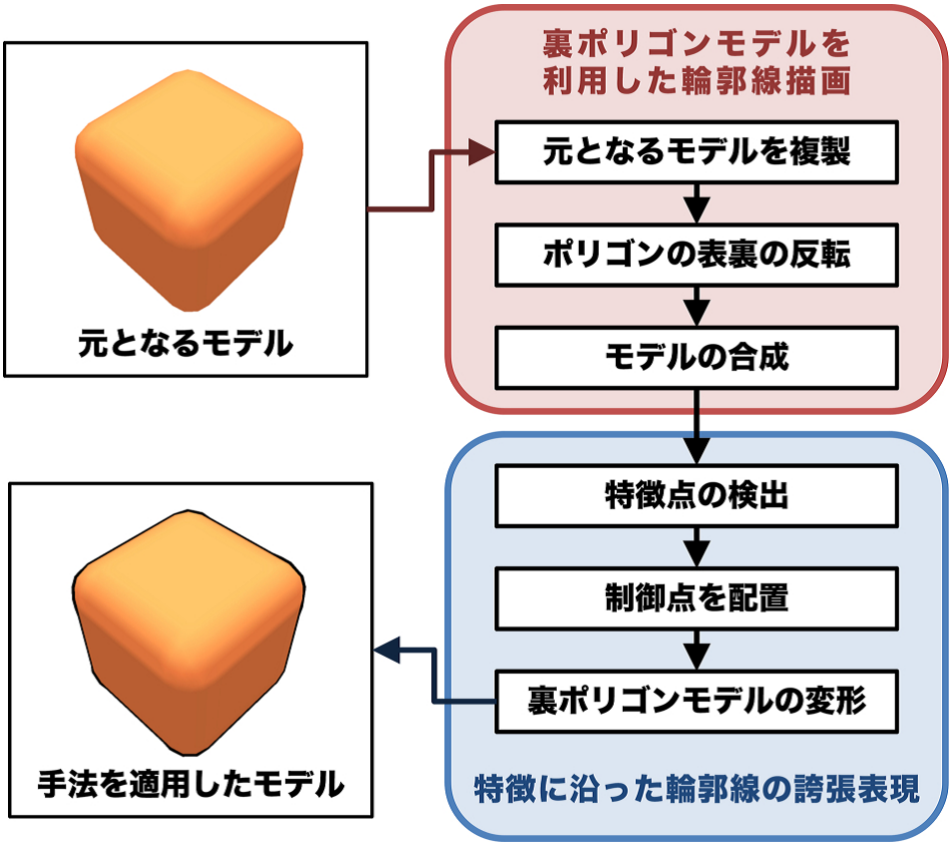


図 3.1: 提案手法の流れ

本章での手法は次の 2 つのステップに大きく分かれている．

1. 裏ポリゴンモデルを利用した輪郭線の描画

2. 裏ポリゴンモデルの変形による輪郭線の誇張表現

まず，ポリゴンの裏面で構成したモデルである，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 3.2.2 節で述べる．次に，モデルの特徴である特徴点の検出を 3.2.3 節で述べ，特徴点と変形を制御する制御点を用いた裏ポリゴンモデルの変形による輪郭線の誇張表現について 3.2.4 節で述べる．また，3.2.5 節では本手法を利用するにあたり，形状を効果的に誇張表現するための調整方法について述べる．

3.2.2 裏ポリゴンモデルを利用した輪郭線描画

本研究では，リアルタイム 3DCG 上で形状に沿った輪郭線の誇張表現を行う．このため輪郭線の表現における基本的な要件として，次の 2 点を考慮することとした．

- 形状に沿う正確な輪郭線
- リアルタイムレンダリングを維持可能な実行速度

これらを踏まえ本研究では，Raskar らのモデルを複製した後に引き伸ばし，ポリゴンの表裏を反転する手法 [24] を用いた．その概略として鈴木らの研究 [39] における手法を次に述べる．

(1) モデルの複製

ベース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．

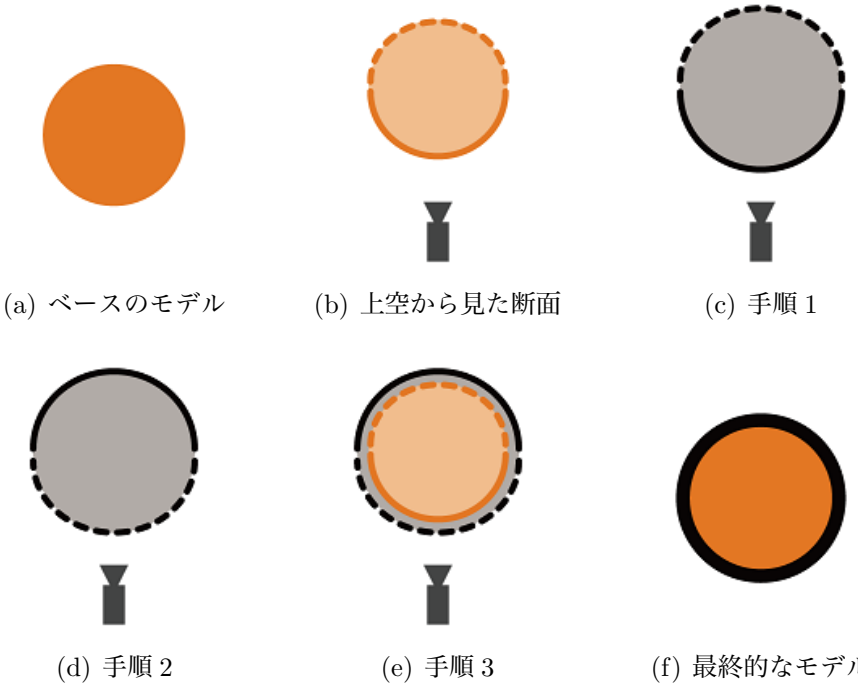
(2) ポリゴンの表裏の反転

拡大したモデルのポリゴンの表裏を反転し，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ようにする．

(3) モデルの合成

拡大し，ポリゴンの表裏を反転したモデルに，ベースとなるモデルを重ね合わせる．

図 3.2 は球体のモデルを利用して，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) ベースのモデル　(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

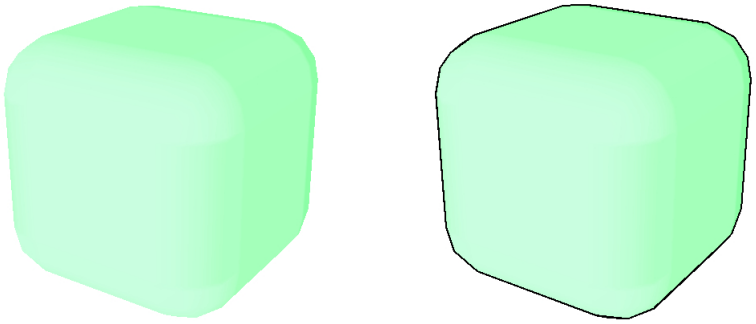
図 3.2: 輪郭線描画の過程

図 3.2(a) と図 3.2(f) は対象となる球体のモデルを正面から見た図である．図 3.2(b)〜図 3.2(e) は，球体の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 3.2(b) はベースとなるモデルを上空から見た時の断面図であり，図 3.2(c) は複製して拡大し，面を黒く塗ったモデル，図 3.2(d) はポリゴンの表裏を反転したモデル，図 3.2(e) はベースのモデルを重ね合わせた図である．この処理を用いた

結果，ベースのモデルからはみ出した部分が表示される．図 3.2(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 3.3 は，以上の輪郭線の描画処理を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) 輪郭線の描画後

図 3.3: 裏ポリゴンを利用した輪郭線

元のモデルである図 3.3(a) に，処理を行った結果が図 3.3(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，単純な処理で均一な輪郭線が描画できる点と，輪郭線となるモデル（以下「裏ポリゴンモデル」）を変形することで輪郭線を容易に変化できる点がある．また，単純に同じモデルを 2 つ重ねて用いるため，形状に沿った正確な輪郭線が描画でき，かつ処理速度も維持できる．本章ではこの手法を用い，リアルタイム 3DCG 上で形状を考慮した輪郭線の誇張表現手法を提案する．

3.2.3　特徴点の検出

本研究では 2.2 節で述べたように，線の強弱は曲線を描く部分を対象としている．モデル上における曲線を描く凹凸部分である，図 2.16 の青い丸，赤い丸で示す箇所のような形状を検出するために，地神らの研究 [40] における，各頂点の法線を利用したモデルの凹凸部分を検出する手法を拡張して用いた．

図 3.4 は，3 次元空間上で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．

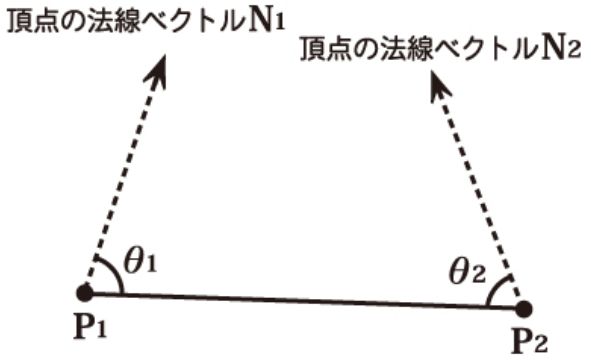


図 3.4: 接続している頂点とその法線ベクトル

モデル上のある頂点座標を $\mathbf{P_1}$ とし，その頂点に隣接し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メッシュ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 3.5 は，頂点の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．

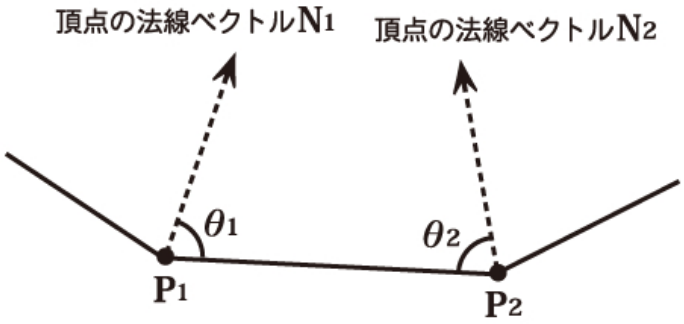


図 3.5: 凹形状のポリゴンの接続

この場合，θ_1 と θ_2 の合計が小さいほど，頂点の法線ベクトルがより向かい合っており，より凹形状に近い状態である．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 3.6 は，頂点の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．

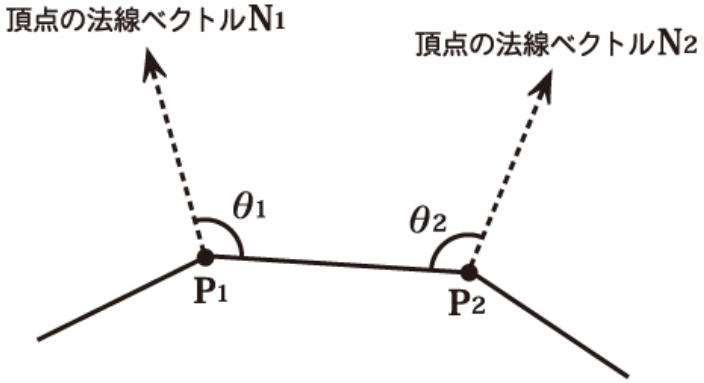


図 3.6: 凸形状のポリゴンの接続

この場合，θ_1 と θ_2 の合計が大きいほど，頂点の法線ベクトルがより反対を向いており，より凸形に近い状態である．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) を満たす隣接する頂点の数が任意の値よりも多い場合，図 2.16 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) を満たす隣接する頂点の数が任意の値よりも多い場合，図 2.16 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．以降，検出した凹凸形状の特徴部分である頂点を「特徴点」と呼ぶこととする．

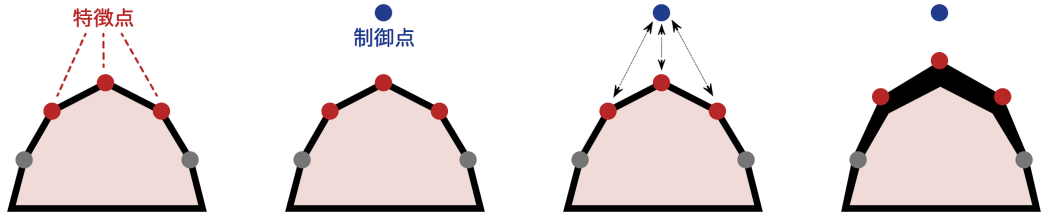
またこのとき，隣接する頂点の数と比較する任意の値（以下「比較値」）により，特徴点となる頂点の検出に変化が生じる．仮に，比較値を大きく設定した場合，隣接する頂点の多くに凹凸形状の特徴がある場合に特徴点となり，より凹凸の変化が顕著な部分が特徴点となる．反対に，比較値を小さく設定した場合，隣接する頂点のいずれかに凹凸形状の特徴がある場合に特徴点となり，凹凸の変化が少なくとも特徴点となる．

よって，より凹凸の変化が大きい箇所のみを特徴点とする場合は，比較値を大きく設定し，全体的に凹凸の変化がある箇所を特徴点とする場合は，比較値を小さく設定する．

3.2.4　制御点を用いた裏ポリゴンモデルの変形

3.2.3 節で検出した特徴点を用い，裏ポリゴンモデルに変形処理を行う．この裏ポリゴンモデルの変化によって，誇張した輪郭線の表現を行う．次の図 3.7 は裏ポリゴンモデルに行う処理の過程を示した図である．なお，図中の赤い点は特徴点

であり，青い点は裏ポリゴンモデルの変形を制御するための点 (以下「制御点」) である．



(a) 輪郭線を描画　(b) 制御点を配置　(c) 関連付け　(d) 特徴点を移動

図 3.7: 制御点を利用した裏ポリゴンモデルの変形

図 3.7 で示す，特徴点と制御点を用いた裏ポリゴンモデルの変形処理を次の順に行う．

(1) 輪郭線を描画

まず，3.2.2 節の手法を利用し，輪郭線となる裏ポリゴンモデルを描画する．また，3.2.3 節の手法を利用し，裏ポリゴンモデルの特徴点を検出する．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 3.7(a) である．

(2) 制御点を配置

次に，モデルの周囲に変形を制御するための制御点を配置する．図 3.7(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，裏ポリゴンモデルの特徴点を移動する方向に手動で配置する．この配置した制御点に対して特徴点となる頂点を移動することで，裏ポリゴンモデルを変形する．

(3) 特徴点と制御点の関連付け

また，特徴点から最も近い座標にある制御点と特徴点を関連付ける．図 3.7(c) は，制御点と特徴点を関連付けを表した図である．頂点数が多く特徴点が多い場合

や，配置する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，次の手順で利用する．

(4) 特徴点を移動

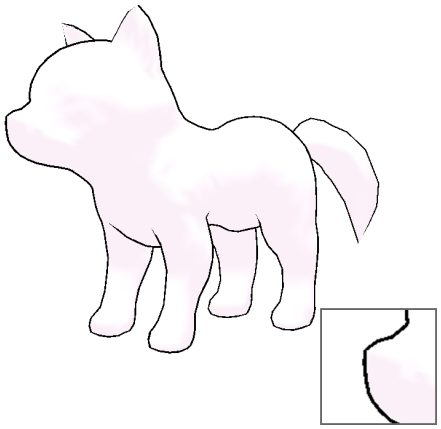
最後に，特徴点と制御点の関連付けから，頂点の法線方向に特徴点を動かすことで輪郭線の誇張表現を行う．図 3.7(d) は，特徴点を移動することでモデルの変形を行った図である．このモデルの変形にあたり，特徴点となる s 個の頂点の位置ベクトルを $\mathbf{x}_i(i = 0, 1, 2, \ldots, s-1)$ とし，制御点となる t 個の点の位置ベクトルを $\mathbf{x}_j(j = 0, 1, 2, \ldots, t-1)$ とする．また，特徴点 $\mathbf{x}_i$ における頂点の法線ベクトルを $\mathbf{n}_i$ とする．このとき，特徴点 $\mathbf{x}_i$ を法線方向に移動した後の位置 $\mathbf{x}_{i,j}$ を次の式 (3.5) で求める．なお，u は任意の定数である．

$$\mathbf{x}_{i,j} = \mathbf{x}_i + \frac{u}{(|\mathbf{x}_i| - |\mathbf{x}_j|)^2} \cdot \mathbf{n}_i \tag{3.5}$$

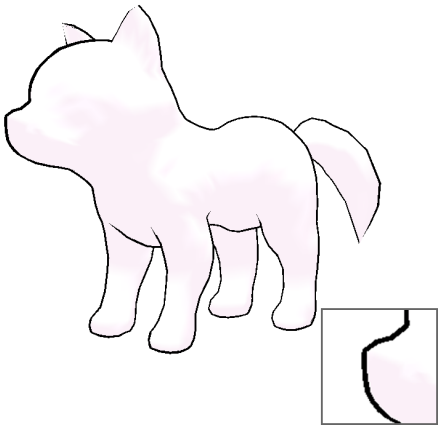
特徴点 $\mathbf{x}_i$ を式 (3.5) で求めた $\mathbf{x}_{i,j}$ に移動することで変形を行う．また，式 (3.5) は関連付けた制御点と特徴点間の距離の 2 乗による反比例式である．このため特徴点は，制御点との距離が近いほどより大きく制御点の方に移動し，距離が遠くなるほど移動距離は短くなる．これにより，制御点に近い特徴点部分を強調し，徐々に強調の度合いが小さくなるような滑らかな表現を行う．

以上の処理を行った結果を図 3.8 に示す．なお，図 3.8(b) と図 3.8(c) の右下の図は同一部分を拡大した図である．

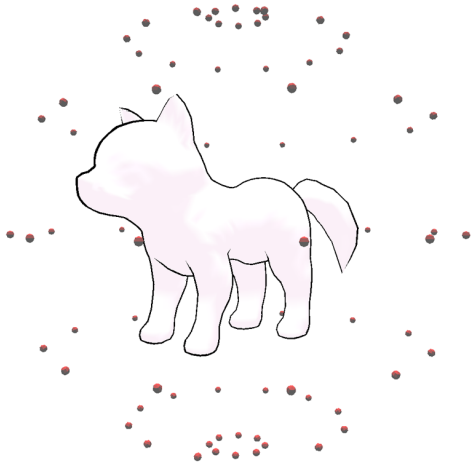
(a) 元のモデル



(b) 均一な太さの輪郭線のモデル



(c) 本手法を用いたモデル



(d) 制御点を表示した状態

図 3.8: 本手法を用いた輪郭線の誇張表現

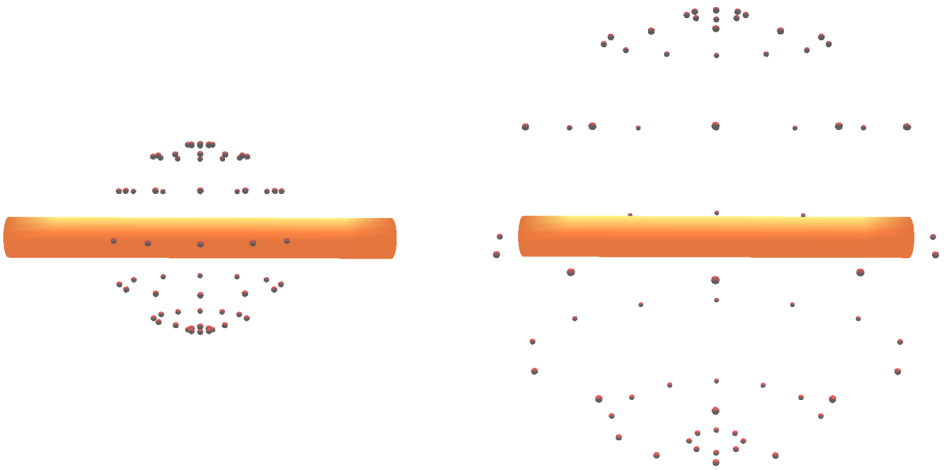
図 3.8(a) は元となるモデルであり，図 3.8(b) は均一な太さの輪郭線を表示したモデルである．図 3.8(c) は本手法を用いた結果であり，図 3.8(d) は図 3.8(c) で用いた制御点を球体のモデルとして可視化したものである．なお，図 3.8(c) における制御点の配置は図 3.8(d) に示す通り，モデルの周囲を球状に囲むように配置した．

図 3.8(b) に示す均一な太さの輪郭線を描画したモデルに対して，本手法を用いたモデルである図 3.8(c) の輪郭線には強弱がついていることが分かる．

3.2.5　モデルの形状に対応した制御点の調整

図 3.8(c) や図 3.8(d) で配置した制御点は，簡易的に一定の条件下で配置した．しかしこの場合，縦長や横長といった極端な形状のモデルを使うと大きくはみ出すことがある．

図 3.9 は，一定の条件下で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．図 3.9(a) は縦幅を，図 3.9(b) は，横幅を基準として配置した場合である．



(a) モデルがはみ出す例　　(b) 大きく囲んだ例

図 3.9: 制御点の配置例

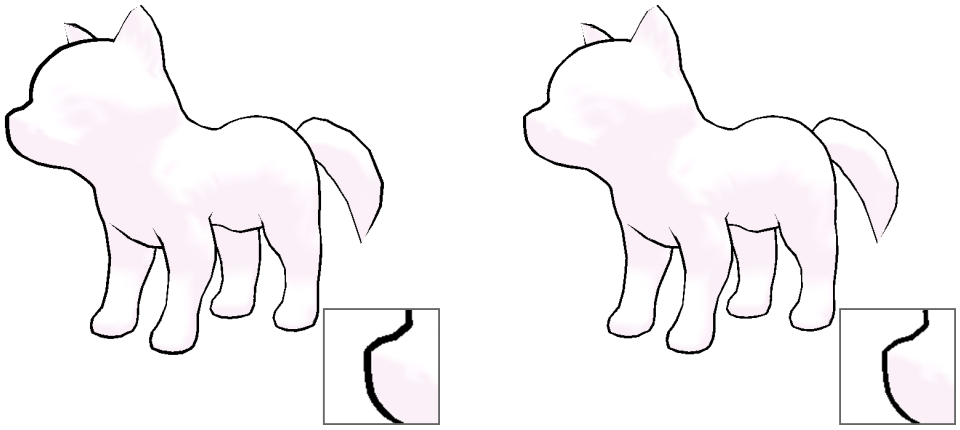
図 3.9(a) は，横幅が足りずにはみ出した状態となっていることが分かる．図 3.9(b) は，モデルの中央付近と制御点の距離が大幅に離れていることが分かる．

図 3.9(a) に示すように，モデルがはみ出すように制御点を配置した場合，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，特徴点が制御点の外側にある場合，制御点との距離が極端に近くなることもあり，予期しない変形が生じる可能性がある．このため，特徴点を意図的に変化できるよう，常に制御点の内側にモデルが存在するような制御点の配置を行う必要がある．

また，図 3.9(b) に示すように，モデルを大きく包み込むように制御点を配置し

た場合，特徴点から最も近い制御点が遠い位置にある可能性がある．3.2.4 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る．大きく球状に包み込んだ場合，特徴点と制御点間の距離が増加するため，特徴点の移動量は減少し，輪郭線の変化は小さくなる．

図 3.10 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．なお，図 3.10 中の輪郭線は変化をより明確とするために，式 (3.5) における裏ポリゴンモデルの移動量を通常よりも大きくした．



(a) 制御点を球状に配置　　(b) 2 倍の距離に配置

図 3.10: 特徴点と制御点間の距離による変化

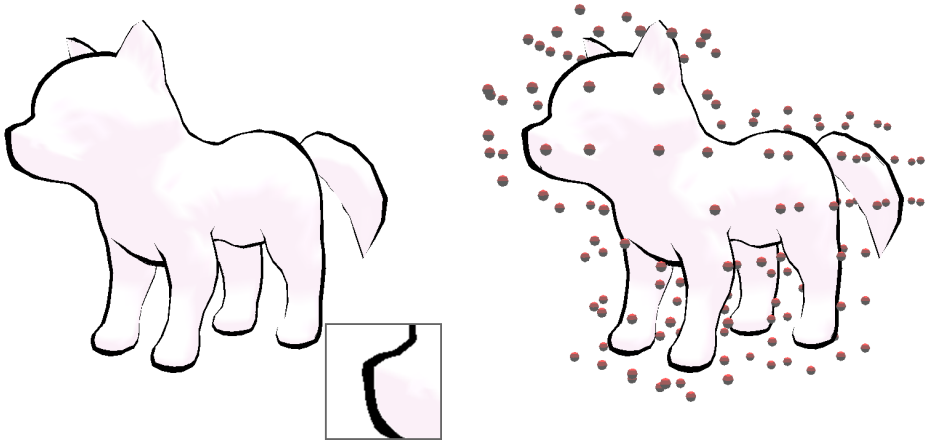
図 3.10(a) と図 3.10(b) はともに，モデルの周囲を球状に包むような図 3.8(d) と同様の制御点の配置をしたものである．ただし，図 3.10(b) の制御点は，図 3.10(a) での配置の 2 倍の距離に配置している．

通常の制御点の配置である図 3.10(a) は，図 3.10(b) に比べ，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，制御点を通常の 2 倍の距離に配置した図 3.10(b) の場合，図 3.10(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くなる場合，本手法を効果的に利用で

きない．よりバランス良く効果的に表現するためには，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となる配置である，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 3.11 は，モデルの形状に沿って制御点を配置した状態を図示したものである．図 3.11(a) は形状に沿って制御点を配置した場合の結果であり，図 3.11(b) は図 3.11(a) で配置した制御点を可視化した図である．



(a) 手動で配置した結果　　(b) 制御点を表示した状態

図 3.11: 形状に沿って制御点を配置した場合

制御点を球状に配置した図 3.8(c) は全体的に膨らんでいる様子に近いが，形状に沿って制御点を配置した場合の図 3.11(a) は，図 3.8(c) よりも線に強弱が出ている．このことより，制御点は球状に配置するなどの簡易的な配置よりも，形状に沿った配置の方がより効果が高いことが分かる．

3.3 検証

3.3.1 効果の検証

本手法を用いた輪郭線の誇張表現について検証する．検証では提案した手法に沿って実装したプログラムを用い，その効果を検証する．検証に用いたプログラムは，グラフィックス API の “OpenGL[41]” をベースとした 3 次元グラフィックスツールキットである “Fine Kernel Tool Kit System[42]” を用いて実装した．

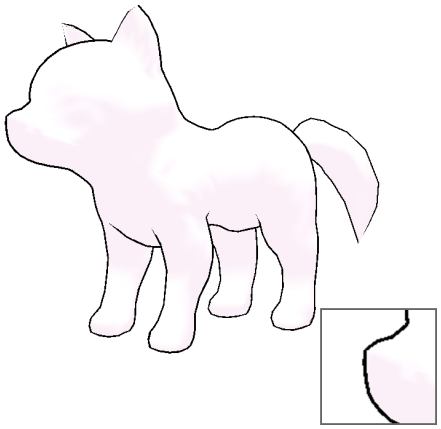
検証に使用した環境を次の表 3.1 に示す．

表 3.1: 実行環境

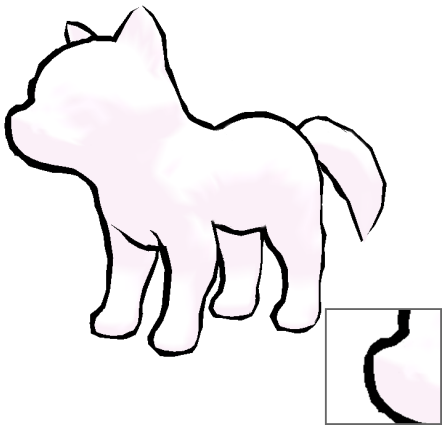
OS	Windows 7 Enterprise 64bit
CPU	Intel Core2 Duo 3.16GHz
メモリ	4.00GB

図 3.12 と図 3.13 は複数の輪郭線表現を行ったモデルの画像である．図 3.12(a)，図 3.13(a) は輪郭線表現をしていないモデル，図 3.12(b)，図 3.13(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 3.12(c)，図 3.13(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 3.12(d)，図 3.13(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，無作為な変化による輪郭線の例 [30] を参考に誇張表現を行った結果を示す．

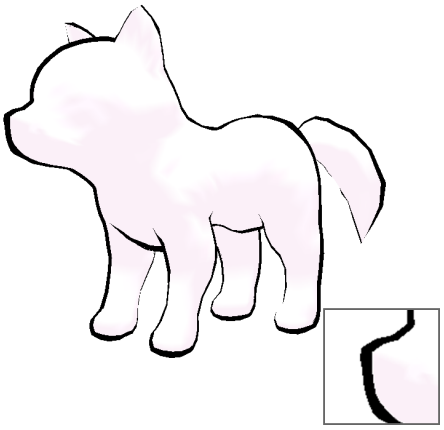
(a) 元のモデル



(b) 均一な太さの輪郭線表現

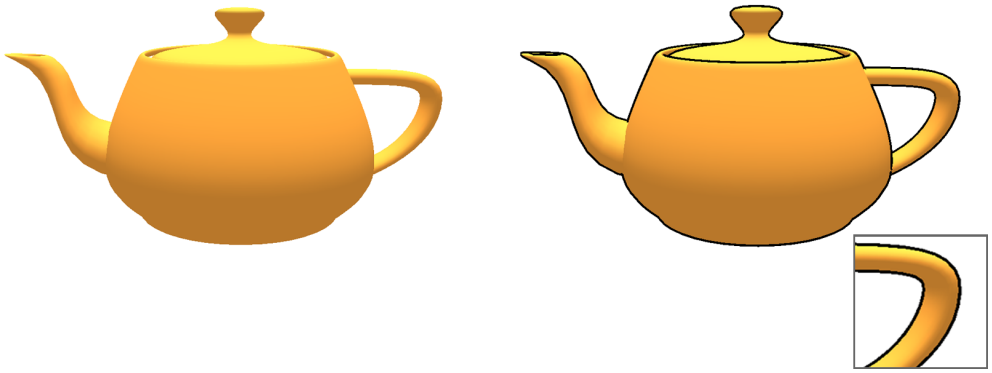


(c) 無作為な線での表現

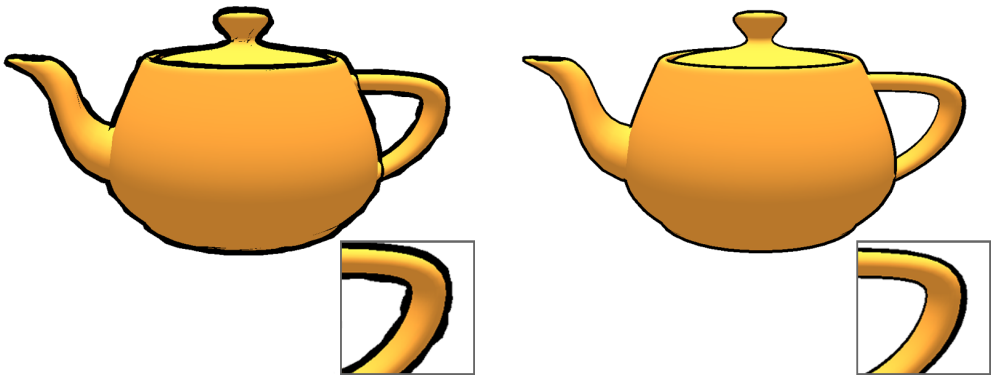


(d) 本手法を用いた表現

図 3.12: 輪郭線表現の比較 (1)



(a) 元のモデル　　(b) 均一な太さの輪郭線表現



(c) 無作為な線での表現　　(d) 本手法を用いた表現

図 3.13: 輪郭線表現の比較 (2)

図 3.12(b) や図 3.13(b) の均一な太さの輪郭線表現に対し，図 3.12(d) や図 3.13(d) の本手法の輪郭線表現は，線に連続した強弱の変化が出ていることが分かる．

また，無作為に線を揺らす表現である図 3.12(c) や図 3.13(c) のような誇張表現は，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．このため，モデルの形状を効果的に表現できていない．これに対し本手法では，図 3.12(d) や図 3.13(d) に示すように，誇張した輪郭線によって，よりモデルの形状の凹凸を

強調するように表現している．

以上のことより，本手法は既存の手法よりも輪郭線や形状をより豊かに表現し，絵を描いた時のような太さの変化のある輪郭線の表現をしているといえる．

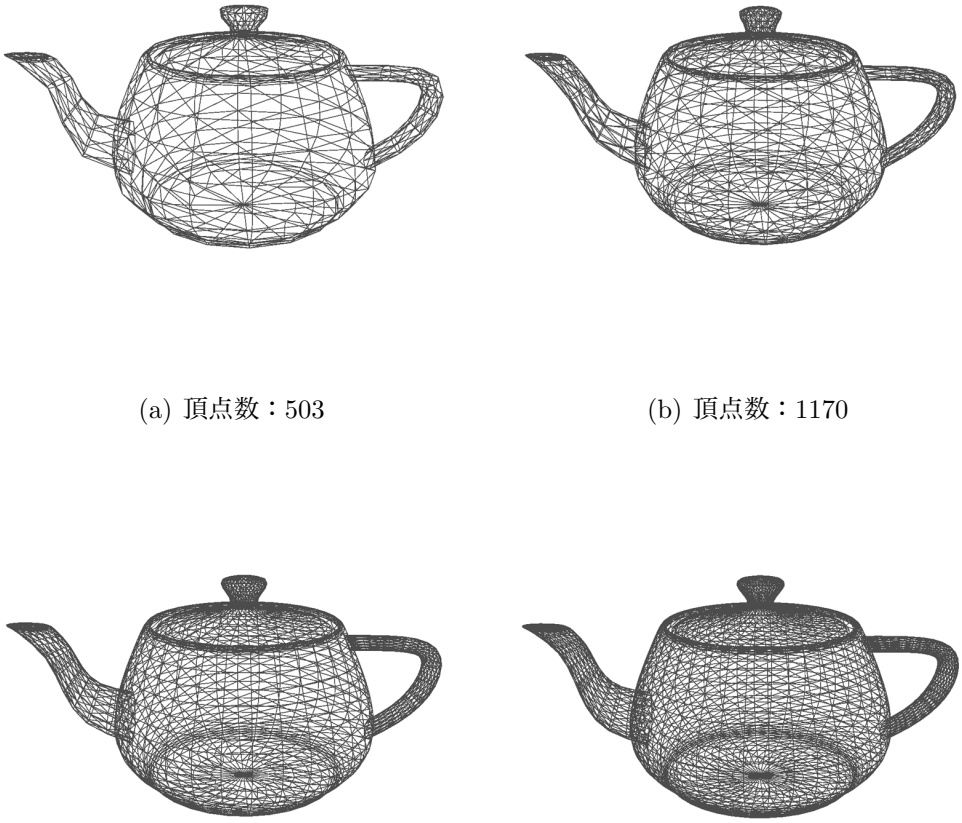
3.3.2 リアルタイム性の検証

本手法のリアルタイム性を検証する．検証には 3.3.1 節と同様に，表 3.1 の環境を用いた．

提案手法では，モデルの頂点を特徴点とし，制御点を用いて頂点を移動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，次の 3 手法において比較し，検証を行うこととした．

1. 3.2.2 節での均一な太さの輪郭線の描画手法
2. 事前計算の 1 回のみで輪郭線誇張のための裏ポリゴンモデルの変形処理を行う手法
3. 毎フレームに輪郭線誇張のための裏ポリゴンモデルの変形処理を行う手法

また検証用のモデルとして，頂点数が異なる同形状のモデルを用意した．検証に利用したモデルをワイヤーフレーム化したものを次の図 3.14 に示す．



(a) 頂点数：503　　(b) 頂点数：1170

(c) 頂点数：2082　　(d) 頂点数：3242

図 3.14: 頂点数が違う同じ形状のモデル

それぞれのモデルの頂点数は，図 3.14(a) は 503 個，図 3.14(b) は 1170 個，図 3.14 は 2082 個，図 3.14 は 3242 個である．

図 3.14 のモデルに対し，次の 2 つの計算方法による誇張表現でのリアルタイム性の検証を行った．

(1) 事前計算による誇張表現

事前計算による誇張表現は，事前に本手法を 1 回適用し，以後本手法を適用せ

ずに描画する方法である．この場合，本手法を適用するのは事前計算の 1 回のみのため，それ以後に新たな特徴点が生じたり，制御点を移動する場合でも，輪郭線の太さが変わることはない．

表 3.2 は，通常の均一な太さの輪郭線表現と事前計算による輪郭線の誇張表現の 1 秒間当たりの描画回数を表したものである．なお，描画速度の単位は frames per second(以下「fps」) である．

表 3.2: 描画速度の測定結果

頂点数	描画速度 (fps)	
	均一な輪郭線	1 回のみの変形
530	653.2	647.0
1170	351.4	340.8
2082	197.8	193.0
3242	74.0	72.6

事前計算よる変形の場合，事前に変形処理を行い，以降は変形処理を行わない．このため，描画の際には均一な太さの輪郭線表現と同様の状態であり，通常の輪郭線処理と同様の描画速度で異なる表現が可能である．

(2) 毎フレームの計算による誇張表現

ゲームなどのコンテンツでは，モーションの変化などによってモデル形状が変化することがある．モデルの形状が変化した場合，特徴点となる頂点が変化したり，特徴点と制御点の位置関係が変化する可能性があり，輪郭線の誇張表現を効果的に表現できない．このため本章では，それらの形状変化を想定して毎フレームに形状変化する場合の検証を行った．

表 3.3 は，事前計算による輪郭線の誇張表現と毎フレームの計算による輪郭線の誇張表現の 1 秒間当たりの描画回数を表したものである．なお，描画速度の単位は表 3.2 と同様に fps である．

表 3.3: 描画速度の測定結果

頂点数	描画速度 (fps)	
	1 回のみの変形	毎フレームの変形
530	647.0	133.0
1170	340.8	54.2
2082	193.0	28.8
3242	72.6	8.0

毎フレームの計算による変形の場合，事前計算による変形よりも描画速度は低下する．しかし，表 3.3 に示すように，頂点数が 2000 個程度までであれば，30fps 程度の描画速度を維持できている．このため，頂点数が一定の範囲内であれば，毎フレームの計算による表現もリアルタイムで実行できる．よって，アニメーションなどによって特徴点や制御点が変化する場合でも，本手法は適用可能であるといえる．

3.4 まとめ

本章の手法では，特徴点と制御点という 2 種類の点群を用いることで，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，評価実験を行った．この結果，次のことが可能となった．

1. 物体の形状を効果的に誇張する表現

イラストやデッサンに用いる，線の強弱による表現に注目し，モデル上で凹凸となる部分を特徴点として検出，制御点によって誇張することにより，対象とする物体の形状に対して，より効果的な輪郭線表現が可能となった．

2. リアルタイムで実行できる処理速度

モデルの形状に沿っており輪郭線が正確であり，かつ処理が単純である裏ポリゴンモデルを利用する輪郭線の表現手法に着目した．これにより，無作為に線を動かさない連続した線の強弱による輪郭線の誇張表現が，リアルタイム 3DCG でのトゥーンレンダリングで実現でき，手描きに近い輪郭線の表現が可能となった．

この手法での課題は，次の通りである．

1. 意図しない変形について

本章の手法では，モデルの頂点の構成から特徴を検出するため，形状によっては意図しない部分を特徴とする可能性がある．このため，今後はモデルの特徴検出をより正確に行えるよう，モデルの曲率から特徴を検出する手法などの新たな手法を検討する必要がある．

2. 毎フレームの計算による誇張表現でのリアルタイム性について

毎フレームの計算による誇張表現に関して，現状では頂点数によるリアルタイム性の制約がある．これに対し，グラフィックスハードウェアである GPU(Graphics Processing Unit) を用いることで，より多くの頂点を処理できるようにすることが望まれる．

3. 制御点の配置について

本章の手法で最適な結果を得るためには，制御点を配置する座標を手動で指定していく必要があり，煩雑な作業である．また，モデルの形状が変化する際には制御点を移動する必要があるため，その都度変更するのも手間がかかってしまう．これに対し，自動的にモデルの周囲に制御点を配置する手法や，形状変形に沿った制御点の配置手法などのより容易に手法を適用できる手法を検討する必要がある．

以上のように，本章の手法では特徴点と制御点を用いることで，輪郭線の誇張表現において一定の効果を得ることができた．しかし，一方で特徴点と制御点を

用いることによる問題点も生じているため，これらの問題を解決する必要がある．特に制御点の配置に関する問題は，多様なモデルを利用したり，またそのモデルの形状が変化したりするゲームなどのコンテンツで用いる際に大きな問題となり得る．このため次章では，本章の問題を踏まえつつ，ゲームなどのコンテンツでより容易に利用するための誇張表現手法を新たに提案する．

第 4 章

3次元モデルの曲率を用いた輪郭線の誇張表現手法

4.1　はじめに

第 3 章の手法では，制御点と特徴点という 2 種類の点群を用いることで輪郭線の誇張表現を実現した．しかし，3.4 節で述べたようないくつかの問題点がある．

本章では，3.4 節で述べた問題点を踏まえつつ，新たにモデルの曲率を用いた輪郭線の誇張表現について述べる．まず，4.2 節で提案手法を述べ，4.3 節で提案した手法を実装したプログラムを用いた検証について述べる．最後に 4.4 節でまとめを述べる．

4.2　曲率を用いた輪郭線の誇張表現手法

4.2.1　手法の概要

図 4.1 は，提案手法の概略を示した図である．

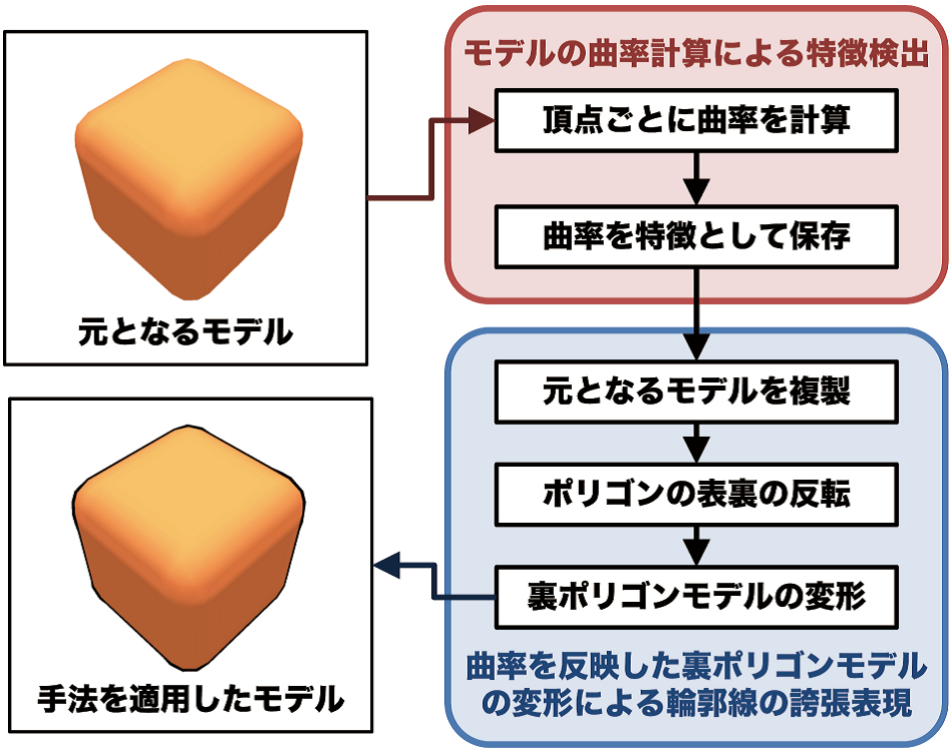


図 4.1: 提案手法の流れ

本章の手法は次の 2 つのステップに分かれている．

1. モデルを構成する面の曲率の計算による特徴の検出

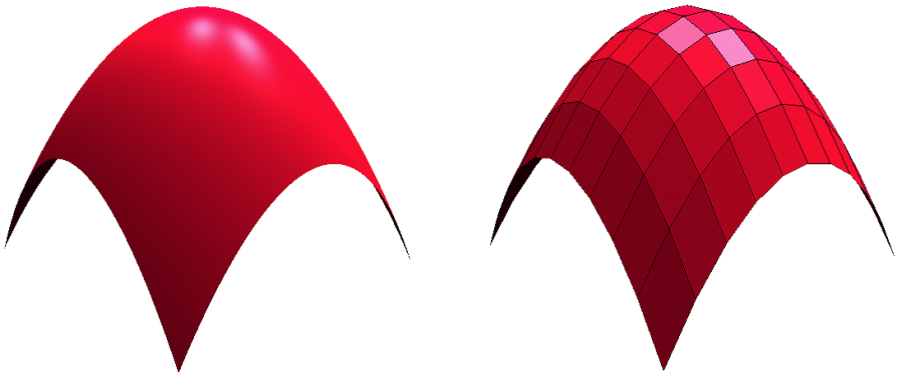
2. 曲率を反映した裏ポリゴンモデルの変形による輪郭線の誇張表現

まず，モデルがどの程度の凹凸であるかの指標である曲率を計算する手法について 4.2.2 節で述べる．次に，モデルの曲率を用いた裏ポリゴンモデルの変形による輪郭線の誇張表現手法について 4.2.3 節で述べる．

4.2.2　3 次元モデルの曲率計算

本章の手法においても 2.2 節で述べたような，モデル上での凹凸形状となる部分を特徴とする．第 3 章の手法では，凹凸形状を構成する頂点を特徴点としたが，本章の手法ではモデルがどれだけ曲がっているかという指標である曲率を用いることとした．

3D モデルに対する曲率算出方法は大きく 2 つに分かれている．ひとつは数式によって表すパラメトリック曲面で構成したモデルの曲率計算方法であり，もうひとつは本研究での対象でもある，多角形で構成したポリゴンメッシュ上でのモデルの曲率計算方法である．図 4.2 にパラメトリック曲面とポリゴンメッシュの例を示す．



(a) パラメトリック曲面の例　　(b) ポリゴンメッシュの例

図 4.2: パラメトリック曲面とポリゴンメッシュの例

図 4.2(a) に示すパラメトリック曲面の場合，曲面を数式で構成しているため，滑らかな面となり曲率を計算するのは容易である．図 4.2(b) に示すポリゴンメッシュの場合は，平面状の多角形を張り合わせた多面体であり，滑らかな面ではないため正確な曲率を計算するのは難しい．しかし，ポリゴンメッシュ上で曲率の近似値を求める手法は多くあり [43][44][45]，ポリゴンメッシュで構成したモデルの曲率の計算を実現している．

本章の手法では，モデル上の各頂点において曲率を計算することで凹凸形状を判別することとし，モデルの特徴を検出する．この頂点の曲率を計算する手法として，Meyer らの手法 [46] での平均曲率を計算する手法を用いた．

図 4.3 はあるポリゴンメッシュ上の頂点 x_i とその周りに隣接する頂点を示したものである．

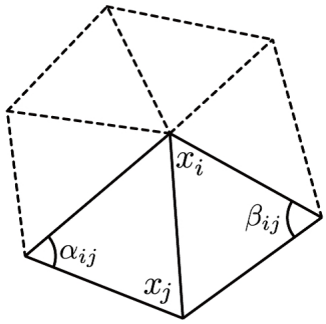


図 4.3: ポリゴンメッシュの例

頂点 $\mathbf{x}_i$ と稜線で結ばれている隣接する頂点 m 個の集合を $N(i)$ とし，$N(i)$ に属する頂点を $\mathbf{x}_j(j = 0, 1, 2, ..., m-1)$ とする．このとき，頂点 $\mathbf{x}_i$ の曲率法線 $K(\mathbf{x}_i)\mathbf{n}_i$ を式 (4.1) に示す．なお，式 (4.1) 式 (4.2) における α_{ij} と β_{ij} は，稜線 $\mathbf{x}_i, \mathbf{x}_j$ を共有する，2 つの 3 角形ポリゴンの向かい合う角度である．

$$K(\mathbf{x}_i)\mathbf{n}_i = \frac{1}{4A(i)} \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{x}_j - \mathbf{x}_i) \tag{4.1}$$

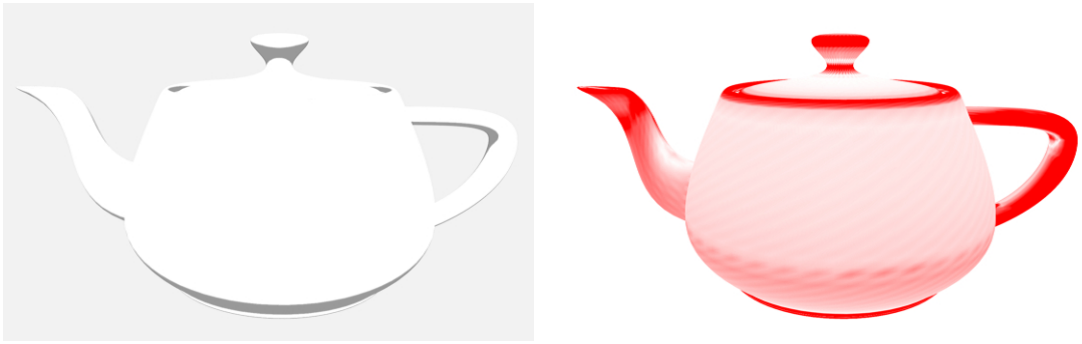
また，点 $\mathbf{x}_i$ を中心としたポリゴンメッシュにおける $\mathbf{x}_i$ のボロノイ面積 $A(i)$ を次の式 (4.2) に示す．ボロノイ面積とは，ポリゴンメッシュ上で最も近い頂点が点 $\mathbf{x}_i$ となるメッシュ上の領域の面積のことである．

$$A(i) = \frac{1}{8} \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij})|\mathbf{x}_i - \mathbf{x}_j|^2 \tag{4.2}$$

各頂点において平均曲率法線 $K(\mathbf{x}_i)\mathbf{n}_i$ を計算し，頂点の平均曲率 $K(\mathbf{x}_i)$ を求める．本章の手法ではこの平均曲率の計算をすべての頂点で行い，モデルの曲率を計算した．

図 4.4 は，図 4.4(a) のモデルに対して式 (4.1) および式 (4.2) の式を用いて計算した曲率 $K(\mathbf{x}_i)$ に対し，色をつけることで変化を可視化したモデルである図 4.4(b)

を示したものである．

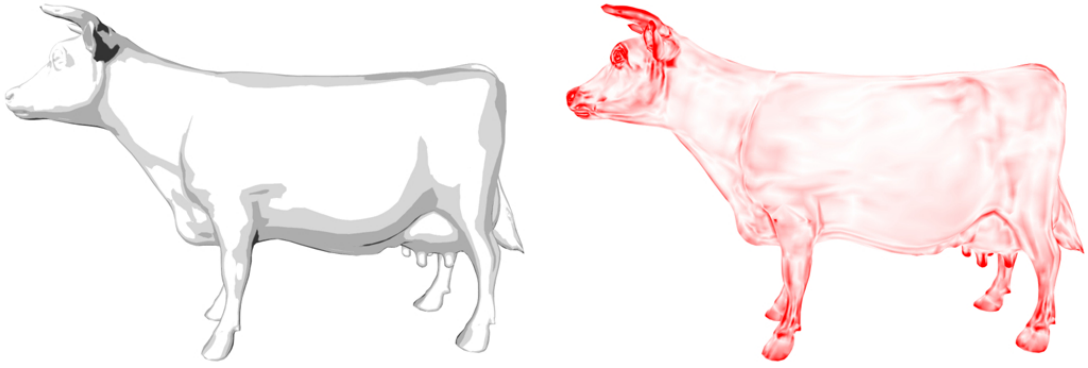


(a) 元となるモデル　　(b) 色づけした曲率の変化

図 4.4: 曲率の変化の様子 (1)

図 4.4(b) の図中で赤色が濃いほど曲率 $K(\mathbf{x}_i)$ が高く，白色が濃いほど曲率 $K(\mathbf{x}_i)$ が低いことを示している．図 4.4(b) から，凹凸形状である箇所は赤色が濃くなっていることが確認できる．

また，図 4.5 は図 4.4 とは違うモデルにおける曲率の変化を示した図である．図 4.5(a) のモデルに対し，曲率の変化を可視化したモデルが図 4.5(b) である．



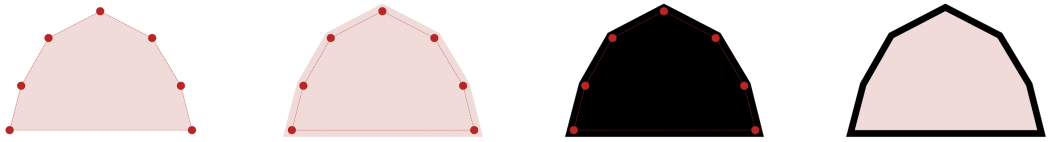
(a) 元となるモデル　　(b) 色づけした曲率の変化

図 4.5: 曲率の変化の様子 (2)

図 4.5(b) も図 4.4(b) と同様に，凹凸形状である箇所は赤色が濃くなっていることが確認できる．

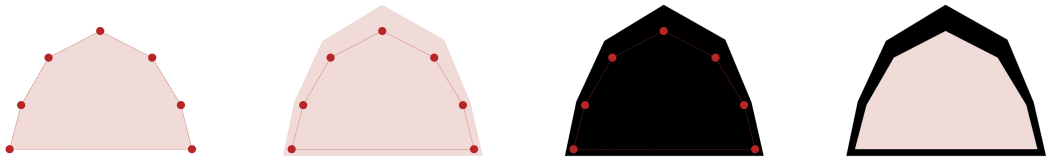
4.2.3　曲率による太さ変化のある輪郭線の描画

4.2.2 節で計算したモデルの頂点ごとの曲率を利用し，輪郭線の誇張表現を行う．本章の手法でも，輪郭線の描画手法は 3.2.2 節での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ととした．次の図 4.6 に，3.2.2 節で述べた手法での図 3.2(c)〜図 3.2(e) での行程を細分化した図を示す．また，次の図 4.7 に，曲率を反映した上での裏ポリゴンモデルの変形過程を示す．



(a) ベースモデルの複製　(b) 輪郭線モデルの拡大　(c) ポリゴン面の反転　(d) ベースモデルと合成

図 4.6: 裏ポリゴンモデルの変形による均一な太さの輪郭線の生成方法



(a) ベースモデルの複製　(b) 曲率を反映した変形　(c) ポリゴン面の反転　(d) ベースモデルと合成

図 4.7: 曲率を反映した変化のある輪郭線の生成方法

図 4.6(a) 図 4.7(a) は輪郭線を描画する対象であるベースのモデルを複製したモデル，図 4.6(b) 図 4.7(b) は複製したモデルの頂点を法線方向に拡大したモデル，図 4.6(c) 図 4.7(c) はポリゴンの表裏を反転して黒く塗りつぶしたモデル，図 4.6(d) 図 4.7(d) はベースのモデルを重ね合わせた図である．

本章の手法では図 4.7 の行程において，図 4.7(b) での法線方向に拡大する際に曲率を反映することで，輪郭線の誇張表現を行うこととし，次の手順で裏ポリゴンモデルの変形処理を行う．

(1) ベースモデルの複製

まず，輪郭線を描画する対象となるベースのモデルを複製する．また，4.2.2 節における式 (4.1) を利用し，モデルの頂点ごとに平均曲率 $K(\mathbf{x}_i)$ を計算する．図 4.7(a) はベースとなるモデルを複製した状態である．

(2) 曲率を反映した変形

次に，複製したモデルの頂点を法線方向に拡大することで輪郭線を生成する．このとき，先に計算した曲率を拡大する値に反映することにより，輪郭線の太さに変化を加える．

計算した曲率から連続した変化にするために，まず，式 (4.1) で計算した曲率 $K(\mathbf{x}_i)$ に任意のしきい値 $s(0 < s \leq 1)$ を設定し，また輪郭線の太さの最大値として任意の値 $t(0 < t)$ を乗算する．その後，しきい値を設定した $K(\mathbf{x}_i)$ に輪郭線の太さの基準となる任意の定数 $u(0 < u)$ を加算し，頂点 $\mathbf{x}_i$ における法線ベクトル $\mathbf{n}_i$ に乗算することで，頂点の移動ベクトル $\mathbf{V}_i$ を設定する．頂点の移動ベクトル $\mathbf{V}_i$ を算出する式を式 (4.3) に示す．

$$\mathbf{V}_i = \begin{cases} \left(\frac{K(\mathbf{x}_i)\,t}{s} + u\right)\mathbf{n}_i & (s > K(\mathbf{x}_i)) \\ (t+u)\,\mathbf{n}_i & (s \leq K(\mathbf{x}_i)) \end{cases} \tag{4.3}$$

頂点 $\mathbf{x}_i$ において $\mathbf{V}_i$ 方向に移動することで，変化のある輪郭線を表現する．

このとき，曲率 $K(\mathbf{x}_i)$ のしきい値である s の値が大きいほど輪郭線の変化の幅が大きくなり，s の値が小さいほど輪郭線の変化の幅が小さくなる．このため，曲率の高い箇所の輪郭線の太さ変化を意図的に大きくする場合には s の値を大きく

し，曲率の高低を満遍なく輪郭線の太さ変化に反映する場合は s の値を小さくすることにより，輪郭線の太さ変化を制御することができる．

また，輪郭線の太さの最大値である t の値が大きいほど太さ変化の変化量が大きくなり，t の値が小さいほど太さ変化の変化量が小さくなる．このため，t の値が大きい場合には曲率が高い箇所と低い箇所での輪郭線の太さの差が大きくなり，t の値が小さい場合には曲率の変化による輪郭線の太さの差は小さくなる．

輪郭線の太さの基準となる u の値は，曲率が 0 となる場合の輪郭線の太さを定義する値であり，u の値が大きいほど輪郭線そのものが太くなり，u の値が小さいほど輪郭線は細くなる．このとき，輪郭線の太さを定義する頂点の移動ベクトル $\mathbf{V}_i$ は $u\,\mathbf{n}_i$ よりも大きくなり，$u\,\mathbf{n}_i$ より小さくなることはない．

図 4.7(b) は，複製したモデルの拡大に曲率を反映した状態である．曲率を反映していない図 4.6(b) と比べ，大きさに変化が出ていることが分かる．

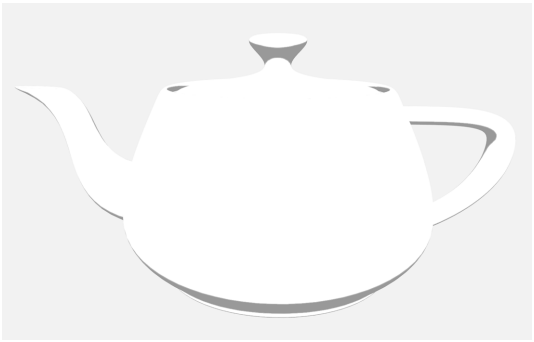
(3) ポリゴン面の反転

複製し，拡大したモデルを輪郭線のモデルにするため，ポリゴンの面を反転し黒く塗りつぶす．図 4.7(c) は，曲率を反映して拡大したモデルのポリゴンの面を反転し，黒く塗りつぶした様子である．

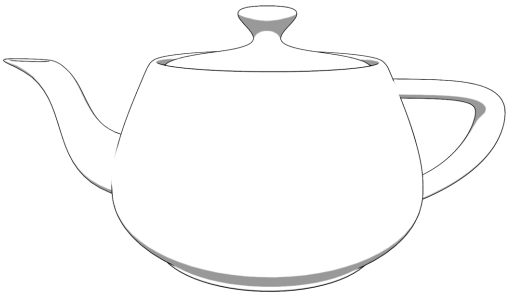
(4) ベースとなるモデルと合成

最後に，複製し拡大したモデルを輪郭線として描画する．図 4.7(d) は，図 4.7(c) のモデルをベースのモデルと合成した様子である．均一な太さの輪郭線を生成した図 4.6(d) と比べ，曲率を反映した図 4.7(d) の方が輪郭線の太さに変化が出ていることが分かる．

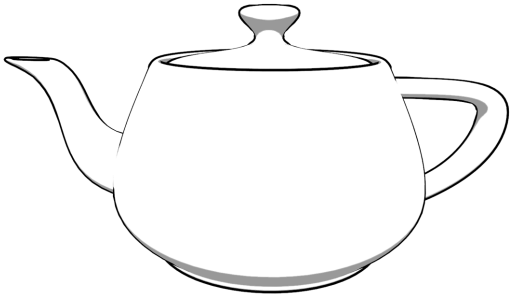
図 4.8 は，以上の輪郭線の描画処理をモデルに適用した結果を示した図である．



(a) 元となるモデル



(b) 均一な太さの輪郭線



(c) 本章の手法の輪郭線

図 4.8: 裏ポリゴンモデルに曲率を反映した輪郭線

ベースとなるモデルである図 4.8(a) に，図 4.6 での処理を行った結果が図 4.8(b) であり，図 4.7 での処理を行った結果が図 4.8(c) である．均一な太さの輪郭線である図 4.8(b) と比べ，本章の手法を用いた図 4.8(c) は輪郭線の太さに変化に差が出ていることが確認できる．

4.3 検証

4.3.1 効果の検証

本章での手法を用いた輪郭線の誇張表現について検証する．検証では提案した手法に沿って実装したプログラムを用い，その効果を検証する．検証に用いたプログラムは，グラフィックス API の “OpenGL[41]” と OpenGL 拡張機能のライブラリである “OpenGL Extension Wrangler Library(GLEW)[47]”，OpenGL を

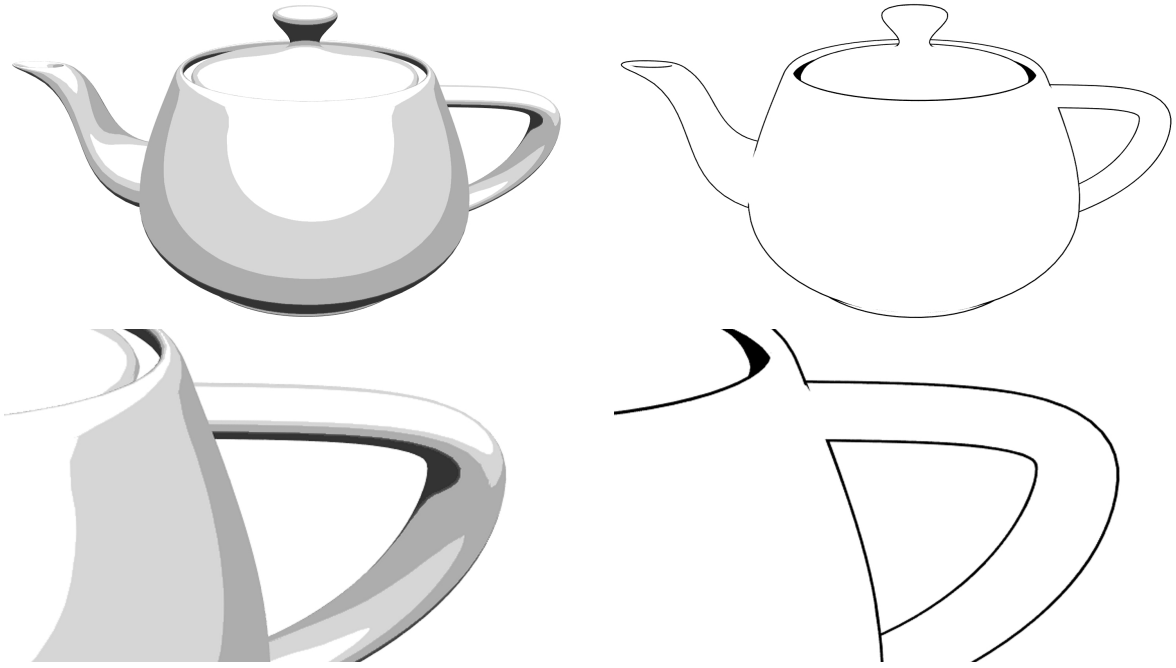
ベースとした 3 次元グラフィックスツールキットである “Fine Kernel Tool Kit System[42]” を用いて実装した．また，シェーダプログラミングには “OpenGL Shading Language(GLSL)[48]” を用いて実装した．

検証に使用した環境を次の表 4.1 に示す．

表 4.1: 実行環境

OS	Windows 7 Enterprise 64bit
CPU	Intel Core i7 3.4GHz
メモリ	8.00GB
GPU	NVIDIA GeForce GTX 560 Ti
解像度	1024 × 768

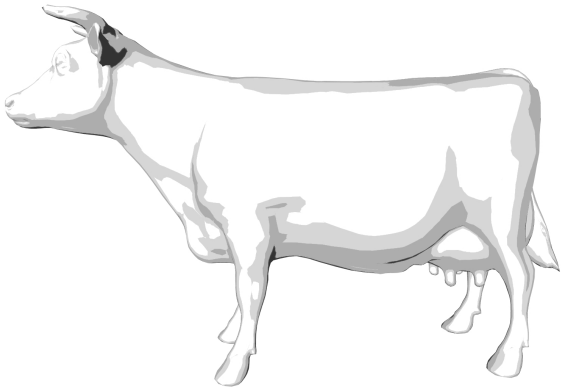
図 4.9 と図 4.10 は複数の輪郭線表現を行ったモデルの画像である．図 4.9(a)，図 4.10(a) は輪郭線表現をしていないモデル，図 4.9(b)，図 4.10(b) が裏ポリゴンモデルを利用した輪郭線表現を行ったモデルである．図 4.9(c)，図 4.10(c) が曲率の大小を色で表現したモデル，図 4.9(d)，図 4.10(d) が本章の手法を用いた輪郭線の誇張表現をしたモデルである．



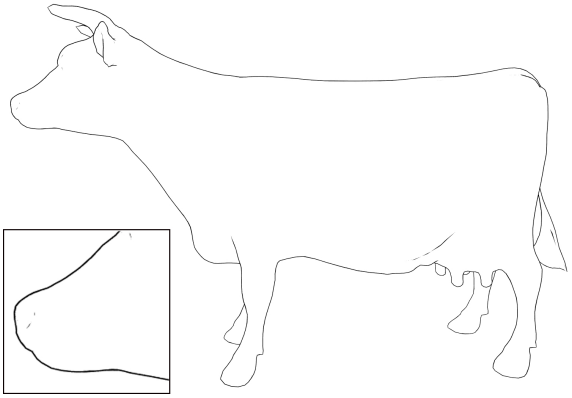
(a) 元となるモデル　(b) 均一な太さの輪郭線表現

(c) 曲率を色で表現　(d) 本章の手法を用いた表現

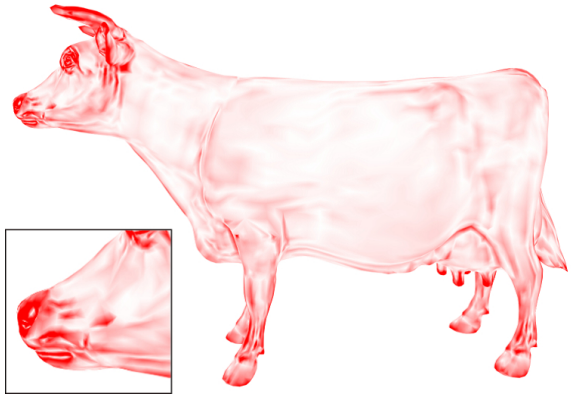
図 4.9: 輪郭線表現の比較 (1)



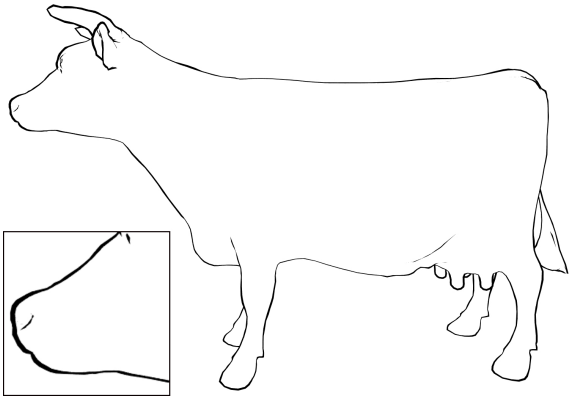
(a) 元となるモデル



(b) 均一な太さの輪郭線表現



(c) 曲率を色で表現



(d) 本章の手法を用いた表現

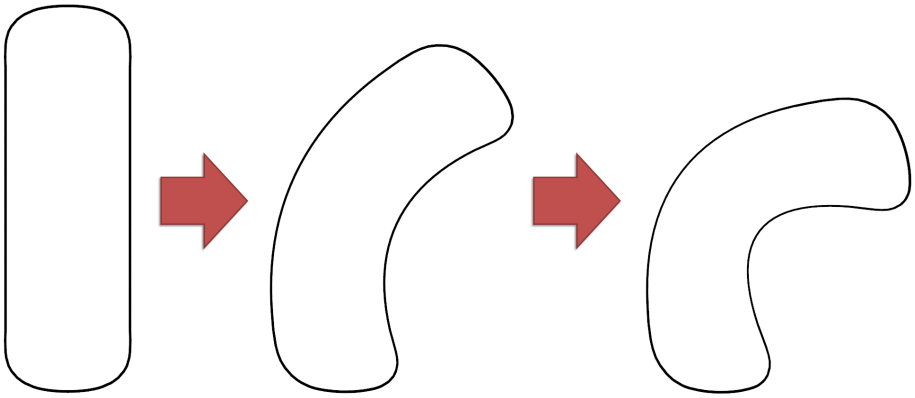
図 4.10: 輪郭線表現の比較 (2)

図 4.9(b) や図 4.10(b) の均一な太さの輪郭線表現に対し，図 4.9(d) や図 4.10(d) の本章の手法の輪郭線表現は，輪郭線に連続した強弱の変化が出ており，誇張した輪郭線によってよりモデルの形状の凹凸を強調するように表現している．また，付録 A に他のモデルを用いた輪郭線表現の比較を行った図を別途示す．

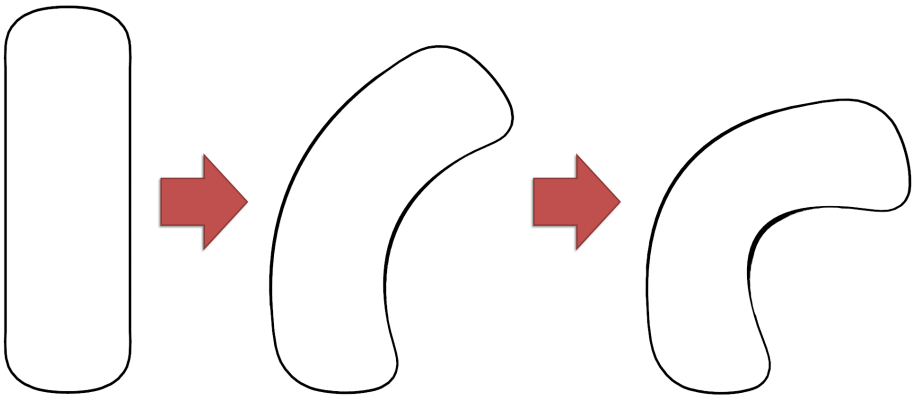
第 3 章の手法ではモデルの形状変形による誇張表現の変化は制御点の移動が煩雑であり不向きであったが，本章の手法ではモデルの曲率によって変形処理を行なっているため，制御点に依存せずに用意に誇張表現の変化が可能である．このため，本章の手法で形状変形に対応した輪郭線の太さ変化表現ができているかをボーンアニメーションを利用し検証を行った．なお，ボーンアニメーションとは，

骨に見立てたボーンで階層構造を作り，ボーンの階層構造とモデルと関連付けることにより，ボーンの動きに沿ったモデルの変形を行うというモデルの変形手法のひとつである [49].

次の図 4.11 は，円柱状のモデルを変形するボーンアニメーションの経過を表したものである．図 4.11(a) は裏ポリゴンモデルを利用した均一な太さの輪郭線表現を行ったモデルのボーンアニメーションの様子であり，図 4.11(b) が本章の手法を用いた輪郭線の誇張表現を行ったモデルのボーンアニメーションの様子である．なお，輪郭線の変化が分かりやすいようにするため，円柱の上部と下部は意図的に輪郭線の変化が生じないように処理を加えた．



(a) 均一な太さの輪郭線でのボーンアニメーション



(b) 本章の手法を適用した輪郭線でのボーンアニメーション

図 4.11: ボーンアニメーション上での輪郭線表現の比較

図 4.9 や図 4.10 と同様に，本章の手法である図 4.11(b) の輪郭線にも連続した強弱の変化が出ていることが分かる．また，変形が進むに応じて曲率が変化しているため，曲がっている箇所の輪郭線の太さも変化しているのが分かる．

以上のことより，本手法は既存の手法よりも輪郭線をより多様に表現し，絵を描いた時のような輪郭線の表現をしているといえる．

4.3.2 リアルタイム性の検証

本手法のリアルタイム性を検証する．検証には 4.3.1 節と同様に，表 4.1 の環境を用いた．

提案手法では，モデルの頂点ごとに曲率を計算し，頂点を移動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，検証を行うこととした．

1. 3.2.2 節での均一な太さの輪郭線の描画手法
2. 事前計算の 1 回のみで輪郭線誇張のための裏ポリゴンモデルの変形処理を行う手法
3. 毎フレームに輪郭線誇張のための裏ポリゴンモデルの変形処理を行う手法
4. 毎フレームにボーンアニメーションによるモデルの変形処理と輪郭線誇張のための裏ポリゴンモデルの変形処理を行う手法

以上の 4 手法を頂点数の違う複数のモデルにおいて行い，次の 3 つの誇張表現上でのリアルタイム性の検証を行った．

(1) 事前計算による誇張表現

事前計算による誇張表現は 3.3.2 節での (1) と同様に，事前に本手法を 1 回適用し，以後本手法を適用せずに描画する方法である．この場合，本手法を適用する

のは事前計算の 1 回のみのため，それ以後にモデルの形状変形などによって曲率が変化した場合でも，輪郭線の太さが変わることはない．

表 4.2 は，通常の均一な太さの輪郭線表現と事前計算による輪郭線の誇張表現の 1 秒間当たりの描画回数を表したものである．なお，描画速度の単位は fps である．

表 4.2: 描画速度の測定結果

頂点数	描画速度 (fps)	
	均一な輪郭線	1 回のみの変形
485	3298.5	3224.8
4250	3023.8	3002.8
15859	2119.6	1833.6
28741	1276.1	1025.4
63576	390.4	306.8

事前計算による変形の場合，事前に裏ポリゴンモデルの変形処理を行い，それ以降は変形処理を行わない．このため，均一な太さの輪郭線表現と同様の描画状態であり，通常の輪郭線処理とほぼ同様の描画速度で異なる表現が可能である．

(2) 毎フレームの計算による誇張表現

毎フレームの計算による誇張表現は 3.3.2 節での (2) と同様にモデルの形状が変化する場合には，モデルの曲率が変化する可能性があり，輪郭線の誇張表現も変化する場合がある．このため本章の手法においても，形状変化を想定した毎フレームでの計算する場合の検証を行った．

表 4.3 は，事前計算による輪郭線の誇張表現と毎フレームの計算による輪郭線の誇張表現の 1 秒間当たりの描画回数を表したものである．

表 4.3: 描画速度の測定結果

頂点数	描画速度 (fps)	
	1 回のみの変形	毎フレームの変形
485	3224.8	3230.6
4250	3002.8	2889.3
15859	1833.6	1500.4
28741	1025.4	863.9
63576	306.8	256.1

毎フレームの計算による変形の場合，事前計算による変形よりも描画速度は低下する．しかし，表 4.3 に示すように，頂点数が多くとも 60fps 以上の描画速度を維持できている．このため，毎フレームに計算しても十分にリアルタイム性を維持することが可能であるといえる．

(3) 毎フレームの計算とボーンアニメーションによる誇張表現

第 3 章の手法では，モデルの形状変形による変化は制御点の操作が煩雑であり不向きであった．しかし，本章の手法ではモデルの曲率によって変形処理を行なっているため，制御点に依存せずに用意に誇張表現の変化が可能である．このため本章では，ボーンアニメーションによって毎フレームに形状変化する場合の検証も行った．

表 4.4 は，事前計算による輪郭線の誇張表現と毎フレームの計算による輪郭線の誇張表現の 1 秒間当たりの描画回数を表したものである．なお表 4.4 は，表 4.2 と表 4.3 と利用したモデルが異なるため，異なる頂点数で検証を行なっている．

表 4.4: 描画速度の測定結果

頂点数	描画速度 (fps)	
	毎フレームの変形	ボーンアニメーションと変形
485	3234.5	3112.5
2419	1653.3	1602.2
10149	731.2	717.2
22226	428.1	416.1

ボーンアニメーションを加えた毎フレームの計算による誇張表現の場合，ボーンアニメーション分の処理が加わるため，毎フレームの誇張表現よりも描画速度は低下する．しかし，頂点数が 20,000 個程度でも 400fps という速度であるため，ボーンアニメーションを用いた変形による表現もリアルタイムで十分に実行できる．よって，ボーンアニメーションなどによって曲率が変化する場合でも，本章の手法は適用可能であるといえる．

4.4 まとめ

本章の手法では，モデルの曲率の変化を用いることで，リアルタイム 3DCG でのトゥーンレンダリングにて形状を誇張する輪郭線の誇張表現を行うための手法を提案し，評価実験を行った．この結果，次のことが可能となった．

(1) 物体の形状を効果的に誇張する表現

第 3 章と同様に，イラストやデッサンに用いる線の強弱による表現に注目した．本章の手法ではモデルの曲率を計算し，モデルの凹凸となる部分を検出することで第 3 章の手法よりもより正確に行い検出した．また，計算したに曲率の変化によって誇張する変化も変えることにより，制御点を配置する手間を軽減した．これにより，第 3 章よりも容易で多様な輪郭線表現が可能となった．

(2) リアルタイムで実行できる処理速度

第3章と同様に，モデルの形状に沿っているため輪郭線が正確であり，かつ処理が単純である裏ポリゴンモデルを利用する輪郭線表現手法に着目した．また，曲率計算と輪郭線の誇張表現の処理をGPU上で行うことで，第3章よりも多くの頂点数での誇張表現が可能となった．これにより，連続した線の強弱による輪郭線の誇張表現がリアルタイム3DCGでのトゥーンレンダリングで実現でき，多様なモデルに対して手描きに近い輪郭線の表現が可能となった．

(3) ボーンアニメーションでの形状変形における誇張表現

本章の手法では，第3章での形状の特徴検出における特徴点検出の問題点と，誇張表現における制御点の配置に関する問題点を解決した．これにより，ボーンアニメーションでの形状の変形にも対応することができ，より多様な輪郭線の誇張表現が可能となった．

この手法での課題は，次の通りである．

(1) 輪郭線の太さの変化の調整について

本章の手法では，輪郭線の太さや変化の幅については任意の値を指定することができるが，第3章の手法のように，制御点を移動することで輪郭線の変化を操作することなどの太さの変化の幅を部分的に変えることは難しい．各頂点ごとの変化の値を変えることにより，輪郭線の太さの変化を操作することは可能であるが，頂点ごとに指定していく必要があるため煩雑である．このため多くの場合では，誇張表現の太さの変化の幅が一定になるため単調な変化となってしまう．これに対し，マテリアルによって変化の幅を変えるような設定や，入力によって太さの変化が調整できるインターフェースなどを加えることにより，容易に太さの変化の幅を変えることができるようにする必要がある．

また本章の手法では，輪郭線の太さ変化の際に均一な太さの輪郭線よりも太くする手法であり，均一な太さの輪郭線の太さよりも細くはならない．このため，均一な太さの輪郭線よりも細くなる変化を取り入れ，太くなるだけではない輪郭線の太さ変化の表現手法を検討していく必要がある．

(2) 曲率計算について

本章の手法での曲率計算には，Meyer らによる平均曲率の計算手法 [46] を用いたが，他のポリゴンメッシュでの平均曲率を計算する手法 [44] や，ガウス曲率など平均曲率以外の曲率を計算する方法 [43][45] もある．このため，今後もリアルタイム性を維持しつつ，より効果が高い曲率計算方法を検討していく必要がある．

以上のように本章の手法ではモデルの曲率の変化を用いることで，輪郭線の誇張表現において第 3 章の手法よりも高い効果を得ることができた．しかし，現状の手法では任意の操作によって太さの幅を部分ごとに変えることや，太さを細くすることができない．このため今後はユーザの入力などによる操作などを可能にすることで，より手描き感のある輪郭線の誇張表現を容易に表現できるようにしたい．

第 5 章

おわりに

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て形状を誇張する輪郭線の誇張表現を行うための手法を提案し，検証を行った．第 3 章の手法では特徴点と制御点を用いて輪郭線の誇張表現を実現し，第 4 章の手法では曲率の変化を用いることで第 3 章の手法の問題点を解決しつつ輪郭線の誇張表現を実現した．表 5.1 は，既存の均一な太さの輪郭線表現と第 3 章の誇張表現手法，第 4 章の誇張表現手法の特徴をまとめたものである．

表 5.1: 手法ごとの特徴の比較

手法	誇張表現	リアルタイム性	誇張表現の操作	形状変形
均一な輪郭線	×	○	×	○
第 3 章の手法	○	○	○	×
第 4 章の手法	○	○	○	○

第 3 章の手法でも第 4 章の手法でも輪郭線の誇張表現をリアルタイムで実現できた．また，第 3 章の誇張表現手法では，制御点の利用によって局所的に変化の幅を変えることが可能であるが，一方で制御点の配置が煩雑なことがあり，形状変形に対応することが難しい．第 4 章の誇張表現手法では，第 3 章の手法を改良し，形状変形に容易に対応でき，任意の定数を変化することで太さの変化を制御できる．しかし局所的に変化を制御したり変化の幅を変えることは難しい．

このため，今後は第 4 章の手法に対して，変化の幅を場所ごとに操作できるようにし，より容易に多彩な太さの変化を表現できるようにしたい．

また本研究では，物体の凹凸やカーブを描く箇所に対して輪郭線の太さを変化することとしたが，実際の 2D アニメーションなどではそれだけでなく，奥行きや光の強さなどの要因でも太さを変えている．今後はこれらの輪郭線の太さが変わる他の要因を踏まえつつ，多彩な輪郭線の表現を実現したい．また，ユーザの入力によって自由に輪郭線を描けるような，よりアーティストにとって直感的な手法を実現することによって，容易に手書き感のある輪郭線をリアルタイム 3DCG

上で行うことができるようにしたい.

なお本研究は，芸術科学会第 9 回 NICOGRAPH 春季大会における “リアルタイム 3DCG における物体の形状を考慮した誇張表現手法の提案 [50]” として発表した内容，芸術科学会第 26 回 NICOGRAPH 秋季大会における “リアルタイム 3DCG における物体の形状を考慮した輪郭線の誇張表現手法の提案 [51]” として発表した内容，Computer Entertainment Developers Conference 2011(CEDEC2011) における “アニメ調輪郭線のリアルタイムレンダリング –連続した太さ変化のある輪郭線表現–[52]” として発表した内容，芸術科学会論文誌第 10 巻 4 号における “リアルタイム 3DCG における物体の形状を考慮した輪郭線の誇張表現手法の提案 [53]” として掲載された内容，SIGGRAPH ASIA 2011 における “Shape Oriented Line Drawing in Real-Time 3DCG[54]” として発表した内容を含む.

謝辞

本論文を締めくくるにあたり，多くの方への感謝の意を表します．

学部１年次から修士課程２年次という６年間もの期間，暖かいご指導ならびに適切なご助言をくださいました，三上浩司先生に多大なる感謝の意を表します．先生から多くのことを教えていただき，多くのことを知り，学ぶことができました．

学部４年次から研究における技術的な側面でのご指導ならびにご助言をくださり，副査をお受けいただいた，渡辺大地先生にも感謝の意を表します．技術だけでなく様々なことを教えていただき，大きく成長することができました．

修士課程において，研究の質や外部発表の側面でのご指導ならびにご助言をくださり，副査をお受けいただいた，近藤邦雄先生にも感謝の意を表します．時に優しく時に鋭い指摘によって，研究や論文の質を大いに高めることができました．

伊藤彰教さん，下田美由紀さんをはじめとしたクリエイティブ・ラボのみなさんや，学部および修士の先輩でもある竹内亮太さん，阿部雅樹さん，武田巧視さん，戀津魁さんに感謝の意を表します．研究だけでなく公私共々に，大変お世話になりました．

近藤・三上研究室の同期として，井上暢三くん，田中希さんや，ケネス・チャンくん，ラウハタイモングコン・ニロートくん，サハ・ジラーユデュンくんをはじめとしたアジア人材の留学生のみなさんにも感謝いたします．みなさんと日々切磋琢磨できたことを本当に誇りに思います．

学部時代は互いに切磋琢磨し，修士時代にも何かと気にかけていただいた，天摩翔くん，佐藤裕瑛くん，瀬戸敦子さんをはじめとしたゲームサイエンスプロジェクト４期生のみなさんにも感謝いたします．同期で唯一の修士はここまで到達することができました．

ゲームサイエンスプロジェクト及び，コンテンツプロデューシングプロジェク

ト，コンテンツプロダクションテクノロジーの先輩や後輩のみなさんにも感謝いたします．また，特に公私共々大変お世話になったゲームサイエンスプロジェクトの小島啓史くんには深く感謝いたします．ありがとう．

その他，各学会でコメントをくださった先生方や学生のみなさん，CEDEC でコメントをいただいたゲーム開発者の方々，記すことのできない大切な方々や，見守ってくれた家族に感謝の意を表します．

また，いついかなる時でも実験の材料となり，時には素晴らしい結果を出し，時には期待を大きく裏切ってくれた，ユタ・ティーポットの 3D モデルにも感謝いたします．ドラゴンよりもバニーよりも良い結果を多く残してくれました．

最後に，本論文に目を通してくださった方に感謝の意を表し，願わくば本論文が少しでも力になることを願っております．

参考文献

[1] Bruce Gooch, Amy Gooch,「Non-Photorealistic Rendering」, AK Peters Ltd, 2001.

[2] Thomas Strothotte, Stefan Schlechtweg, 「Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation」, Morgan Kaufmann, 2002.

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

[4] Thomas Luft, Oliver Deussen, “Interactive Watercolor Animations”, Proc. PacificGraphics 2005, pp.7-5, 2005.

[5] Thomas Luft, Oliver Deussen, “Real-Time Watercolor Illustrations of Plants Using a Blurred Depth Test”, Proc. NPAR 2006, pp.11-22, 2006.

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

[7] Aaron Hertzmann, “Non-Photorealistic Rendering and the Science of Art”, Proc. NPAR 2010, pp.147-157, 2010.

[8] Philippe Decaudin, “Cartoon-looking rendering of 3D-scenes”, Research Report, 1996.

[9] Adam Lake, Carl Marshall, Mark Harris, Marc Blackstein, “Stylized rendering techniques for scalable real-time 3D animation”, Proc. NPAR 2000, pp.13-20, 2000.

[10] 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.

[11] David Bremer, John F. Hughes,“Rapid Approximate Silhouette Rendering Of Implicit Surfaces ”, Proc. Implicit Surfaces 1998, 1998.

[12] Pierre Benard, Forrester Cole, Aleksey Golovinskiy, Adam Finkelstein, “Self-Similar Texture for Coherent Line Stylization”, Proc. NPAR 2010, pp.91-97, 2010.

[13] Stefan S Bert, Bert Schonwalder, Lars Schumann, Thomas Strothotte, “Surfaces To Lines: Rendering Rich Line Drawings”, Proc. WSCG 1998, pp. 354-361, 1998.

[14] AL Guptill, SE Meyer, 「鉛筆で描く」, マール社, pp.33-43, 1977.

[15] AL Guptill, SE Meyer, 「ペンで描く」, マール社, pp.48-56, 1976.

[16] Takafumi Saito, Tokiichiro Takahashi, “Comprehensible Rendering of 3-D Shapes”, Proc. SIGGRAPH 1990, pp.197-206, 1990.

[17] Drew Card, Jason L. Mitchell, “Non-Photorealistic Rendering with Pixel and Vertex Shaders”, ShaderX: Vertex and Pixel Shader Tips and Tricks, pp.319-333, 2002.

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

[19] 望月義典, 近藤邦雄, “再分割曲面の輪郭線抽出描画手法”, 情報処理学会論文誌, Vol.41, No.3, pp.601-607, 2000.

[20] 望月義典, 近藤邦雄, “形状特徴表現のためのエッジ強調描画手法”, 情報処理学会論文誌, Vol.40, No.3, pp.1148-1155, 1999.

[21] 金子満, “次世代アニメーション制作システムに関する研究”, 学位論文, 東京工業大学, 1996.

[22] 望月義典, 近藤邦雄, 佐藤尚, 島田静雄, “形状理解を助けるためのカラー画像の強調表現手法”, 情報処理学会研究報告 グラフィクスと CAD 研究会報告, Vol. 95, No.47, pp.17-22, 1995

[23] Jarek R. Rossignac, Maarten van Emmerik, “Hidden contours on a frame-buffer”, Proc. Eurographics 1992, 1992.

[24] Ramesh Raskar, Michael F Cohen, “Image Precision Silhouette Edges”, Proc. I3D 1999, pp.135-140, 1999.

[25] Ramesh Raskar, “Hardware Support for Non-photorealistic Rendering”, Proc. HWWS 2001, pp.41-47, 2001.

[26] Bruce Gooch, Peter-Pike J. Sloan, Amy Gooch, Peter Shirley, Richard Riesenfeld, “Interactive technical illustration”, Proc. I3D 1999, pp.31-38, 1999.

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

[28] Todd Goodwin, Ian Vollick, Aaron Hertzmann,“Isophote Distance: A Shading Approach to Artistic Stroke Thickness”, Proc. NPAR 2007, pp.53-62, 2007.

[29] “大神”, CAPCOM CO.,LED., カプコン, 2006, 2008.

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

[31] 村上恭子, 鶴野玲治, “顔料及び支持体の特性を考慮したパステル画風レンダリング”, 芸術科学会論文誌, Vol.1, No.2, pp.89-96, 2002.

[32] 川嵜敬二, 中丸幸治, 大野義夫, “NPR におけるストローク方向の決定と水墨画調レンダリングへの適用”, 芸術科学会論文誌, Vol.3, No.4, pp.235-243, 2004.

[33] Michael A. Kowalski, Lee Markosian, J. D. Northrup, Lubomir Bourdev, Ronen Barzel, Loring S. Holden, John F. Hughes, “Art-based rendering of fur, grass, and trees”, Proc. SIGGRAPH 1999, pp.433-438, 1999.

[34] J. D. Northrup, Lee Markosian, “Artistic silhouettes: a hybrid approach”, Proc. NPAR 2000, pp.31-37, 2000.

[35] H Lee, S Kwon, S Lee, “Real-time pencil rendering”, Proc. NPAR 2006, pp.37-45, 2006.

[36] Mario Costa Sousa, John W. Buchanan, “Computer-Generated Graphite Pencil Rendering of 3D Polygonal Models”, Proc. Eurographics 1999, pp.195-208, 1999.

[37] Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, Adam Finkelstein, “WYSIWYG: NPR Drawing Strokes Directly on 3D Models”, Proc. SIGGRAPH 2002, pp.755-762, 2002.

[38] Robert D. Kalnins, Philip L. Davidson, Lee Markosian, Adam Finkelstein, “Coherent Stylized Silhouettes”, Proc. SIGGRAPH 2003, pp.856-861, 2003.

[39] 鈴木隼人, 渡辺大地, “リアルタイム 3DCG における米国漫画調レンダリングの開発”, 情報処理学会 第 66 回 全国大会 講演論文集, 4N-9, pp.185-186, 2004.

[40] 地神知哉, 渡辺大地, “簡略化表現を用いた，アニメ調 3DCG の視認性向上手法”, 第 7 回 情報科学技術フォーラム 講演論文集, I-054, pp.307-308, 2008.

[41] Silicon Graphics International Corp, “OpenGL”, `http://www.opengl.org/`.

[42] Fine Kernel Project, “Fine Kernel ToolKit System”, http://fktoolkit.sourceforge.jp/.

[43] C.R.Calladine, “Gaussian curvature and shell structures”, The Mathematics of Surfaces, Oxford Universlty Press, pp.179-196, 1986.

[44] Jack Goldfeather, Victoria Interrante, “A novel cubic-order algorithm for approximating principal direction vectors”, ACM Transactions on Graphics, Vol.23, pp.45-63, 2004.

[45] Wesley Griffin, Yu Wang, David Berrios, Marc Olano, “GPU curvature estimation on deformable meshes”, Proc. I3D 2011, pp.159-166, 2011.

[46] M Meyer, M Desbrun, P Schroder, A H. Barr,“Discrete Differential-Geometry Operators for Triangulated 2-Manifolds”, Visualization and mathematics, Vol.3, No.3, pp.34-57, 2002.

[47] Milan Ikits, Marcelo Magallon, “The OpenGL Extension Wrangler Library”, http://glew.sourceforge.net/.

[48] The Khronos Group Inc, “The OpenGL Shading Language”, http://www.opengl.org/documentation/glsl/.

[49] CG–ARTS 協会, 「ディジタル映像表現 改訂版」,CG–ARTS 協会, pp.158-159, 2006.

[50] 松尾隆志, 三上浩司, “リアルタイム 3DCG における物体の形状を考慮した誇張表現手法の提案”, 第 9 回 NICOGRAPH 春期大会 ポスターセッション, P04, 2010.

[51] 松尾隆志, 三上浩司, 渡辺大地, 近藤邦雄, “リアルタイム 3DCG における物体の形状を考慮した輪郭線の誇張表現手法の提案”, 第 26 回 NICOGRAPH 秋季大会 論文集, II-2, 2010.

[52] 松尾隆志, 三上浩司, 渡辺大地, 近藤邦雄, “アニメ調輪郭線のリアルタイムレンダリング –連続した太さ変化のある輪郭線表現–”, Computer Entertainment Developers Conference 2011(CEDEC2011), インタラクティブセッション, P0155, 2011.

[53] 松尾隆志, 三上浩司, 渡辺大地, 近藤邦雄, “リアルタイム 3DCG における物体の形状を考慮した輪郭線の誇張表現手法の提案”, 芸術科学会論文誌, Vol.10, No.4, 2011.

[54] Takashi Matsuo, Koji Mikami, Taichi Watanabe, Kunio Kondo, “Shape Oriented Line Drawing in Real-Time 3DCG”, Proc. SIGGRAPH ASIA 2011 Posters, No.45, 2011.

発表論文

松尾隆志, 三上浩司, “リアルタイム 3DCG における物体の形状を考慮した誇張表現手法の提案”, 第 9 回 NICOGRAPH 春期大会 ポスターセッション, P04, 2010.

松尾隆志, 三上浩司, 渡辺大地, 近藤邦雄, “リアルタイム 3DCG における物体の形状を考慮した輪郭線の誇張表現手法の提案”, 第 26 回 NICOGRAPH 秋季大会 論文集, II-2, 2010.

松尾隆志, 三上浩司, 渡辺大地, 近藤邦雄, “アニメ調輪郭線のリアルタイムレンダリング –連続した太さ変化のある輪郭線表現–”, Computer Entertainment Developers Conference 2011(CEDEC2011), インタラクティブセッション, P0155, 2011.

松尾隆志, 三上浩司, 渡辺大地, 近藤邦雄, “リアルタイム 3DCG における物体の形状を考慮した輪郭線の誇張表現手法の提案”, 芸術科学会論文誌, Vol.10, No.4, 2011.

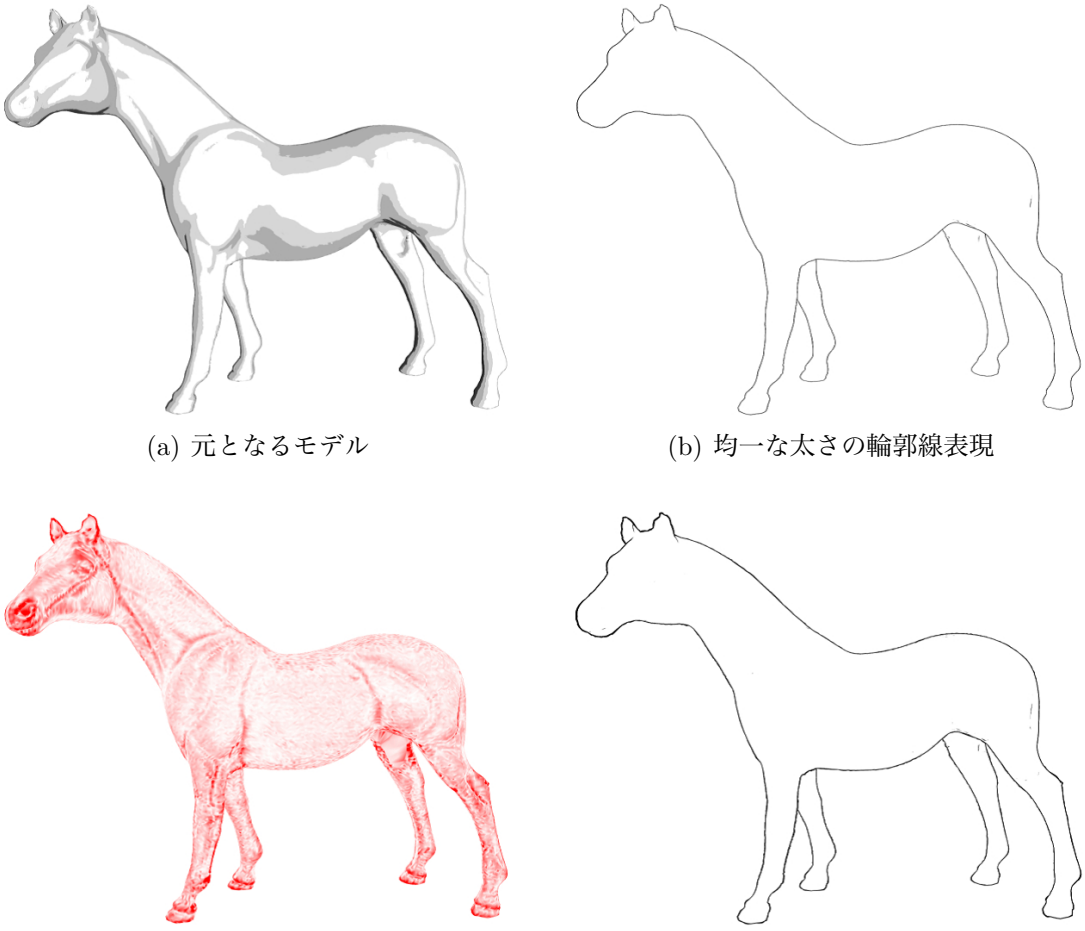
Takashi Matsuo, Koji Mikami, Taichi Watanabe, Kunio Kondo, “Shape Oriented Line Drawing in Real-Time 3DCG”, Proc. SIGGRAPH ASIA 2011 Posters, No.45,

2011.

付録 A

提案手法を用いた例

第 4 章で提案した手法を用い検証した他の結果を示す．なお，図 A.1 と図 A.2 は複数の輪郭線表現を行ったモデルの画像である．図 A.1(a)，図 A.2(a) は輪郭線表現をしていないモデル，図 A.1(b)，図 A.2(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 A.1(c)，図 A.2(c) が曲率の大小を色で表現したモデル，図 A.1(d)，図 A.2(d) が第 4 章の手法を用いた輪郭線の誇張表現をしたモデルである．



(a) 元となるモデル　(b) 均一な太さの輪郭線表現

(c) 曲率を色で表現　(d) 第 4 章の手法を用いた表現

図 A.1: 輪郭線表現の比較 (1)

(a) 元となるモデル

(b) 均一な太さの輪郭線表現

(c) 曲率を色で表現

(d) 第 4 章の手法を用いた表現

図 A.2: 輪郭線表現の比較 (2)