博士論文　　平成 24 年度 (2012)

ストローク履歴を活用した
3次元形状モデリング手法の研究

主査　近藤 邦雄
指導教員　渡辺 大地

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

竹内 亮太

目 次

第 1 章　序論 **1**
1.1 研究背景 2
1.2 メディアとしての 3DCG 4
1.3 本研究の目的 5
1.4 本論文の構成 7
1.5 本論文の用語 8

第 2 章　3DCG モデリング技術と
モデリングインタフェースの研究事例 **9**
2.1 3DCG のデータ表現形式 10
2.1.1 ポリゴン 10
2.1.2 パラメトリック曲面 11
2.1.3 CSG 集合 11
2.1.4 陰関数曲面 11
2.1.5 ボリュームデータ 12
2.1.6 点群 12
2.2 モデリングシステムとインタフェース 13
2.2.1 ヒストリーベースモデラー 13
2.2.2 ダイレクトモデラー 14
2.2.3 スケッチインタフェース 14
2.2.4 スカルプトモデラー 15
2.2.5 シミュレーション・プロシージャルベースモデラー 16
2.3 本研究の位置付け 16

第 3 章　点群とボクセルを併用した
形状試作モデリングシステム **19**
3.1 はじめに 20
3.2 本システムの概要 22
3.3 二重データ構造 25
3.4 画像のシルエットに基づく形状生成 26
3.5 形状編集 28
3.5.1 編集位置の計算 29

3.5.2 陰関数曲面による切削操作 30
3.5.3 凸包形状による切削 32
3.5.4 盛り付け処理 33
3.6 実装と検証 34
3.7 考察 36
3.7.1 点群による形状表現 36
3.7.2 処理速度の検証 37
3.8 まとめ 38

第 4 章 陰関数集合によるストロークベースのモデリングシステム 39
4.1 はじめに 40
4.2 陰関数によるスカルプトモデリングの実現 42
4.2.1 形状表現に用いるデータ構造 42
4.2.2 ストロークを単位とする履歴管理とリターゲット処理 45
4.3 ヒストリーツリーインタフェースの実装 48
4.4 検証 50
4.5 まとめ 53

第 5 章 モデリングシステムの実装と運用 54
5.1 はじめに 55
5.2 モデリングシステムの機能 56
5.3 点群の生成とレンダリングアルゴリズム 61
5.4 モデリングシステムによる作例 62
5.5 モデリングシステムで既存のモデルを編集した例 65
5.6 まとめ 71

第 6 章 結論 74
6.1 研究のまとめと成果 75
6.2 今後の課題と展開 76

謝辞 78

参考文献 81

発表実績 95

第 1 章

序論

1.1 研究背景

コンピュータ技術の発達と普及により，映像やゲームなどのコンテンツを制作する上でコンピュータグラフィクス [1, 2] は無くてはならない技術分野となった．その中でも 3 次元コンピュータグラフィクス (以下，3DCG) 技術 [3] は，用途が多岐にわたることもあり，著しい進歩を遂げている．3DCG をコンテンツ制作に用いることは多くのメリットをもたらす．まず，人物や物体を立体的に表現することができるため，2 次元上での作図や描画よりも，立体感や臨場感において有利である．これを応用して，現実には困難であったり不可能であるシーンを再現することもできる．また，カメラワークやライティングの調整によって，一度制作したデータに編集を加えて再利用することが容易な点も特徴であると言える．

3DCG 技術の分野は，大別すると 3 つに分類することができる．それは，形状表現のためのデータ構造 [4, 5] 及び造形編集手法を構成する「モデリング」，形状の回転・平行移動や変形を行う「アニメーション」，最終的に画面出力を行う「レンダリング」の 3 分野である．アニメーション技術は形状のデータ構造に大きく依存するためモデリングの範疇に含む場合もあるが，近年はよりダイナミックな映像表現のために別個の技術として扱うことが多くなってきた．

しかし，技術の発達によって各分野で行う作業や扱う技術は複雑化の一途を辿っている．レンダリング分野の技術発達は，処理速度の向上や新たな表現の幅を広げることに繋がり，アニメーションにおいてもそれは同様である．その分，モデリングの作業においては進歩した技術に対応するため，制作上求められるデータの規模や品質が上昇し，必要な作業工程が増加している．モデリング分野の技術発達は，レンダリング技術がポリゴンとテクスチャというモデルの構成要素にハードウェアが特化して進化したため，造形作業がこのデータ構造に合わせたものになっている．インタフェースの改良などはされているものの，本質的なモデリング操作はポリゴンを主としたデータ構造に，モデリングツールのユーザが大きく歩み寄らざるを得ないのが現状である．

本研究は，このようにユーザの入力を既存のデータ構造へ無理矢理あてはめることを良しとせず，新たなアプローチを模索するところに端を発している．すなわち，データ構造に合わせたインタフェースではなく，人間が一番入力しやすく，ディジタルならでは利便性の高い操作が可能であることを第一義とし，その操作を実現するために適したデータ構造を提唱する．

既存の 3DCG は，形状の特徴点を頂点として持ち，その接続関係によって面を構成するポリゴンモデルを用いるのが一般的である．形状の特徴点を直接指定せずに，制御点の移動によって凹凸を自由に変形できるパラメトリック曲面も用いることが多いが，最終的に描画を行う際にはポリゴンに変換する．すなわち，リアルタイム 3DCG においてはポリゴンモデルの使用は暗黙の内に強制されてきたものだと言える．

一方で，2次元上でのドローイングにおいては，パラメトリック曲線や頂点と直線によるベクターベースの描画方法と共に，ピクセルの色値を直接操作することで描画を行うラスターベースのペイントツールも用いられてきた．ベクターベースは解像度の影響を受けず，レンダリング先の解像度に適した画像を必要に応じて生成するのには有用なアーキテクチャである．しかし，人間の手によるドローイングを最終的には数式によって近似するということでもある．このため，幾何学的なデザインの表現手段としては優秀であっても，例えば水彩画や油絵のような微細なタッチが芸術性を生み出すような作品分野においては，ユーザの入力をベクターベースに無理矢理当てはめることになる．これはユーザの詳細な入力精度をそぎ落とすことに他ならない．ラスターベースのアーキテクチャは，一定の解像度でユーザの入力をサンプリングすることによって，解像度による制約は受けるものの，入力のニュアンスをピクセル単位で保持することができる．このため，2次元上のドローイングでは，ベクターベースとラスターベースのアーキテクチャを使い分けるのがごく当たり前になっている．

翻って，3DCG においてはベクターベースのアプローチしか現実的な解が存在しない．2次元のピクセルに相当するものとして，3軸方向に格子状のセルを並べ

たボクセルが存在するが，データ量が膨大に膨れあがるため，アーティスティックなユーザの入力を受け止めるだけの解像度を確保するのが困難であると共に，リアルタイムなレンダリングで扱うことも難しい．

また，既存のモデリングツールは編集形状の扱いのみに集中するあまり，ユーザが入力した操作の履歴を保持しきれない編集処理になってしまっている．このため，既存のポリゴンモデラーやスカルプトモデラーでは，アンドゥやリドゥなどの履歴操作をデータのスナップショットによって実現している．編集形状が複雑になるほど形状データの容量も大きなものになってしまうため，アンドゥで遡れる履歴はメモリ上で保持できるデータ容量によって制限を受けてしまう．

旧来のCADソフトにおいては，モデリング自体をユーザが入力したコマンドを全て保持することによる，ヒストリーベースの手法を用いたものが多い．しかし，CADソフトにおけるコマンドは幾何的な形状生成に適したものがほとんどで，スカルプトモデラーのような有機的な形状を生成するのには適さない．

このように，3次元形状モデリング技術は利用分野毎に別個の発展を遂げているものの，それぞれの分野毎に新たな問題点を抱えているのが現状である．

1.2　メディアとしての3DCG

コンピュータ上で図形を表現する技術は1963年頃には存在していたと言われている．しかし，出力が平面の画面であることから，2次元図形と3次元図形では，インタフェース改善の進歩の度合いに差が生じている．その要因として，画面をそのままキャンパスとして扱える2次元画像とは異なり，3次元図形はカメラによって定義する2次元平面のスクリーンへ射影する必要がある．2次元画像の編集ツールは画像処理技術の発達と，マウスやペンタブレットなどのポインティングデバイスの普及により，画面領域を仮想の画材として扱うことができるシステムが多く開発されるようになった．しかし，3次元図形の編集は画面に射影する時点で次元が縮退することから，どうしても仮想空間を覗き込む違和感を払拭することができない．これは，立体映像装置や3次元入力装置のブレイクスルーが無い限り，

根本的な解決は困難であろう．

スカルプトモデラーの誕生は，この問題をある程度解決することに成功している．ペイント動作を彫刻動作になぞらえることで，ドローイング感覚のモデリングシステムを実現した．これにより，形状表面の微細な凹凸を画面上に描き込む感覚で造形することが可能になった．しかし，ポリゴンの再分割を繰り返すためにメッシュの構造が汚いものになること，形状の変形処理がデータを上書きするものであるため，編集履歴の保持が困難であることが問題として挙げられる．つまり，現実の造形に近い操作感覚を得られる反面，ディジタルならではの編集の利便性を両立させることができていないと言える．また，ディティールの描き込みには強い反面，土台となる形状は別途ポリゴンモデラーやスプラインモデラーで生成するという工程が必要となる場合が多い．これもモデリングの敷居を上げてしまう要因となっている．

総じて，3DCG というメディアは使い手であるユーザと，処理を行うコンピュータ側のアーキテクチャの間で両極端な振れ幅のもと生み出されているのが現状であると言える．今こそ，この振れ幅の間でちょうどいい「メディア」となるべきモデリング技術を生み出す必要がある．

1.3 本研究の目的

本研究の目的は，よりよい形状モデリングインタフェースを実現することにある．本論文においては，その実現のために具体的に 2 つの要素から改善を図った．

1 つは，ベクトルベースに依らない形状表現の実現である．既存のポリゴンベースでの形状表現では，スカルプトによる形状変形に際してポリゴンの細分割を行う必要がある．ユーザーの入力が微細になればなるほど，細分割によって生成するポリゴン数は増加していく．ポリゴンデータは頂点間の接続情報を保持する必要があり，複雑な形状の細分割を繰り返し行うと位相構造の再構築に膨大な処理コストを要する．

そこで本研究では，点群というデータ構造に注目した．点群はポリゴンとは異

なり，頂点間の接続を考慮する必要が無く，ただそこに存在するだけの構成要素である．点群の増減は，ポリゴンの増減よりも小さな処理コストで実現できるため，ユーザの入力に対する快適なインタラクションを実現できると考えた．点群を用いることで，ボクセルとは異なるアプローチから，3DCG においてラスター形式として扱うことが可能なデータ構造を提唱する．これにより，ユーザの入力をベクターベースで近似せず，解像度フリーで保持することが可能になる．

もう 1 つは，ユーザの入力履歴を重要なデータとして取り扱う手法の確立である．履歴を完全に保持することができれば，アンドゥ・リドゥを制限なく行えることに加えて，編集手順の途中キャンセルや，入力操作を再利用可能な環境が構築できる．スカルプトモデリングはユーザの入力を反映する手法としては優秀な反面，編集操作が不可逆であるため，履歴操作の実現にはスナップショットを用いる．そうすると当然，コンピュータが扱えるメモリ容量以上のスナップショットは保持できないため，遡れる履歴には制限が生じる．形状モデリングに限らず，履歴を遡ったり，途中の操作のみをキャンセルしたり，分岐した履歴を保持したいと言った需要は存在するため，これらが 3 次元形状モデリングで実現できれば有用であると考えられる．

そこで，ユーザの入力をストローク単位で保存することにより，全履歴を保持したヒストリーベースのスカルプトモデリングを提案する．点群によって形状表面を表現することに加えて，ストロークの入力を 3 次元空間に投影して CSG 集合として扱うことにより，形状の内外情報を保持したソリッドなモデリングを実現した．履歴をストローク単位で保存しているため，形状が変化していたり，視点が変化している場合でも再度投影を行うことで，ユーザの入力を異なる状況でも最大限活かせるようにした．

以上の目的を達成したモデリングシステムを開発し，実際に造形作業を行うことで，本研究が提案する各手法の有用性を検証した．

1.4　本論文の構成

本論文は全6章から成る．図 1.1 は，各章の関連を示すものである．

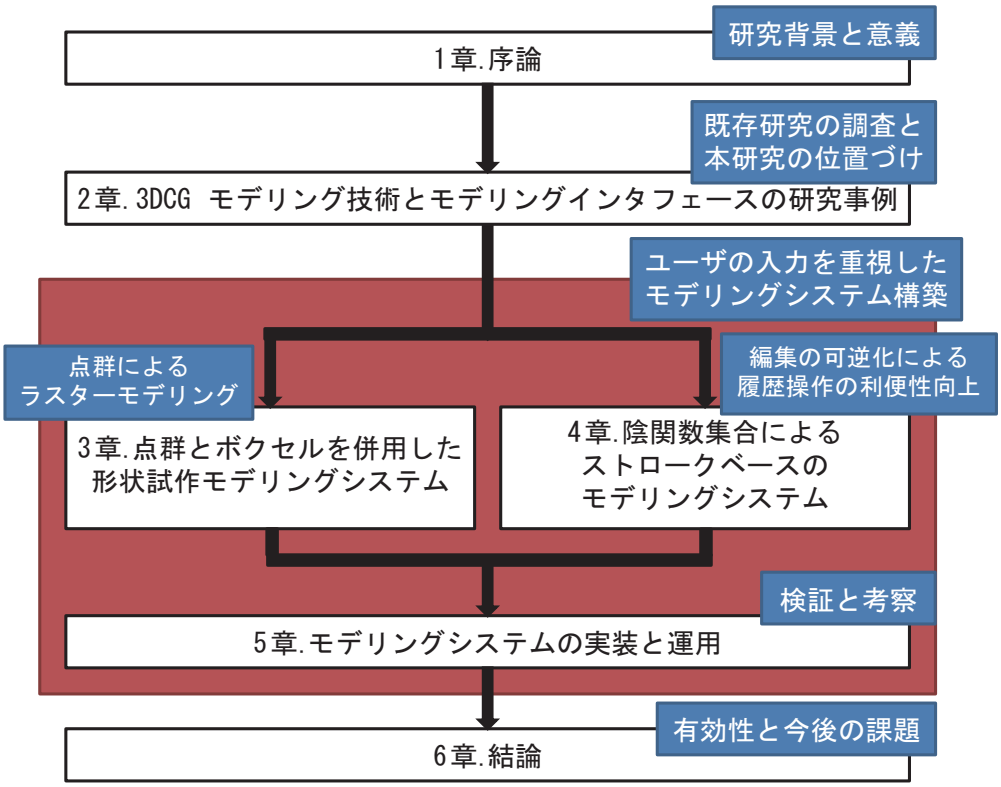


図 1.1: 本論文の章構成

第 2 章では既存の 3 次元データ構造やモデリングシステム，及びそれらに関わる研究についてレビューを行う．しかるのちに，本研究の立ち位置と意義を明確にする．第 3 章は，点群を用いたモデリングシステムを構築し，形状のシルエット画像を入力することによるプロトタイピングに適した手法を提案する．これにより，画像をガイドとしつつ，ユーザーの入力は解像度の影響を受けずに自由な切削操作が可能な技術を確立する．第 4 章は，切削操作の履歴をストロークベースで完全に保存することで，無制限のアンドゥ・リドゥと，履歴の途中キャンセルやリピートが可能なモデリングシステムを構築する．第 5 章は，第 3 章と第 4 章で述

べた手法をもとにモデリングシステムを構築し，運用した結果によって本研究における提案手法の有用性を検証する．最後に，第 6 章にて本論文の結論をまとめ，今後の展望を述べる．

1.5 本論文の用語

以下に，本論文で用いる用語及び表記について解説する．

$\mathbf{p}$	3 次元空間中の任意の点を表す．
$\mathbf{n}_p$	3 次元空間中の任意の点が持つ法線ベクトルを表す．
$\mathbf{c}_p$	3 次元空間中の任意の点が持つ色情報 (RGB) を表す．
$\mathbf{I}(u, v)$	画像上の任意の点 (u, v) が持つ色情報 (RGB) を表す．
α	陰関数曲面上の任意の点を表す．
β	凸包形状の表面上の任意の点を表す．
$\mathbf{a} \cdot \mathbf{b}$	ベクトル $\mathbf{a}$ と $\mathbf{b}$ の内積を表す．
$\mathbf{a} \times \mathbf{b}$	ベクトル $\mathbf{a}$ と $\mathbf{b}$ の外積を表す．

第 2 章

3DCG モデリング技術と
モデリングインタフェースの研究事例

本章では，CG の源流である CAD の設計システムで採用しているモデリング技術から，近年一般的になったスカルプトモデリングに至るまで，個々のシステムで用いているデータ形式とインタフェースをレビューする．その上で，本研究の立ち位置と意義を明確にする．

2.1 3DCG のデータ表現形式

2.1.1 ポリゴン

ポリゴンは立体形状の表面を表す，一番シンプルでよく用いられているデータ構造である．形状の特徴を構成する頂点と，その接続関係を保持することで三角形を構成し，その集合によって形状を構成する．頂点座標を行列によって変換することで，形状全体の移動・回転・拡大縮小やカメラの視点変更に対応できるため，GPU はポリゴンをレンダリングすることに特化した発達を遂げた．このため，現状のハードウェアではリアルタイムレンダリングに最も適したデータ構造と言える．反面，形状の特徴を頂点の配置によって決定するため，ユーザは頂点移動やポリゴンの再分割といったデータ構造を考慮した操作を行う必要があり，独特の感覚が要求される．Cohen ら [6] は，少ないポリゴンで微細な形状の特徴を表現する手法を提案した．これを元に，バンプマッピングや変位マッピングといった手法 [7, 8] が提案され，GPU はポリゴンとテクスチャを組み合わせて扱う用途に最適化して発達していくことになる．Muller ら [9] は，ポリゴンで表現した物体が物理的な挙動によって変形する手法を提案した．Nealen ら [10] は，曲線の入力によってその意図に適切なポリゴンメッシュを生成する手法を提案した．Yang ら [11] は，メッシュ境界面をスムースに接続するためのアルゴリズムを提案した．Schmidt ら [12] は，メッシュのペーストを容易に実現するためのインタフェースを提案した．Nesme ら [13] は，ポリゴンモデルを柔物体としてアニメーションさせるための変形手法を提案した．

2.1.2 パラメトリック曲面

パラメトリック曲面とは，2次元上のパラメトリック曲線を3次元に拡張し，形状表面を表したデータ構造である．パラメトリック曲面の特徴としては，制御点を操作することで，ポリゴンよりも簡易にユーザが望む曲面を生成できることや，適切な拘束条件を設定することで，連続性を保持できることが挙げられる．レンダリングに際してはポリゴンに変換することが多いが，シェーダを利用して直接レンダリングする手法も提案されている．しかし角張った形状には向かない点や，数式で表現しきれない微細なディティールを表現できない点が問題として存在する．一般的に用いられる曲面として NURBS[14] がある．Cashman ら [15] は，細分割曲面を NURBS 曲面に変換して扱う手法を提案している．Schollmeyer ら [16] は，NURBS 曲面を GPU 上で直接トリミングする手法を提案している．

2.1.3 CSG 集合

CSG 集合とは，内外領域を表す関数の集合演算によって形状を表現するデータ構造である．あくまで内外情報のみを取り扱うデータ形式であるため，表面は必要に応じてサンプリングして利用することになる．このサンプリング精度によって，形状の精細さと処理速度のバランスを取ることができる．以前は関数の個数が増加することがコンピュータの記憶容量を圧迫していたことから，複雑な形状の表現には適さないと思われていたが，現在の計算機性能ならば十分な運用が可能であると見込める点が，本研究の着眼点でもある．実用的に運用する際には後述する陰関数曲面を利用することが多い．Requicha ら [17, 18] は，CSG 集合を用いたブーリアン演算によるソリッドモデリングを提案した．

2.1.4 陰関数曲面

陰関数曲面は，複数の関数によって構成した形状をよりスムーズに扱うためのデータ構造である．大竹ら [19, 20] が提案した，MPU 法や SLIM 曲面が代表的な

手法である．可視化するためには等値面をサンプリングし，ポリゴンなどに変換する必要があるが，金井ら [21] は，GPU のシェーダ機能を利用して陰関数曲面をポリゴン化せずに直接描画する手法を提案している．柴田ら [22] は，陰関数を用いた形状のモデリングとモーフィングを行う手法を提案している．Yuan ら [23] は，陰関数をオブジェクトの内部構造を精細に表現するために，階層構造の陰関数表現を提案している．Stam ら [24] は，陰関数曲面の移流を算出して表面を流れるようなアニメーション生成に有効な手法を提案している．松宮ら [25, 26] は，パーティクルシステムと陰関数曲面を用いた仮想粘土細工シミュレーションを実現している．

2.1.5 ボリュームデータ

ボリュームデータは，2 次元画像のピクセルを 3 次元に拡張した，3 次元におけるラスターとでも言うべきデータ形式のことを指す．格子状に並んだ 1 つのセルをボクセルと呼ぶことから，ボクセルデータと呼ぶことも多い [27]．計測データで用いられることが多かったが，近年はボクセルベースのモデラーも登場してきた．内外情報を完璧に保持し，ベクターベースではなく，サンプリングした入力データを直接的に反映できるのは大きなメリットである．しかしながら，大規模かつ高精細なデータを保持するには解像度を上げる必要があり，指数関数的に容量が爆発するのが問題点と言える．ボクセルベースのモデラーとしては，Bærentzen ら [28, 29] の手法や，Böonning ら [30] の手法や，Prior[31] の手法がある．Martin ら [32] や Laurentini[33] は，物体を複数視点から撮影した画像を入力することで，ボリュームデータとしての形状復元を行う手法を提案した．

2.1.6 点群

点群は，3 次元空間中に離散的に分布した点の集合によって形状を表現するデータ形式である．頂点間の接続情報を考慮しなくて良いため，3 次元レンジスキャナによる計測データとして用いることが多い．Avron ら [34] は，スキャンデータから

復元した形状データからシャープさが損なわれる点に対する解決手法を示している．Li ら [35] は，3D スキャナから得た点群データに対し，規則性や対称性を抽出して正確な形状復元を行う手法を提案している．渡邉ら [36] は，スキャンデータとして取得した点群をスクリーン平面にプロットする際に画像処理を施し，芸術的な特徴を捉えた表現を用いたウォークスルーコンテンツを提案している．また，点群ベースで柔物体の変形や融解をアニメーション表現する手法として，Müller ら [37] や Gerszewski ら [38] の提案がある．点群を使ったモデリングシステムとしては，Reuter ら [39] の手法や，Zwicker ら [40] の手法や，Pauly ら [41] の手法がある．点群を直接的に可視化する手法として，Katz ら [42] の手法がある．Pfister ら [43] は，点群の個々の点に対して法線ベクトルと擬似的な領域を持たせて，表面を表すプリミティブとしての利用を提案した．これを利用して，Adams ら [44] はブーリアンモデリングを実現した．Adamson ら [45] は，個々の点を形状全体の流れに沿って変形させることで，スムースな表面形状を表現する手法を提案した．Guennebaud ら [46] は，代数曲面を点群で表現する手法を提案した．Alexa ら [47] は，エルミート曲線に基づいた点群サーフェス表現を提案した．

2.2 モデリングシステムとインタフェース

2.2.1 ヒストリーベースモデラー

ヒストリーベースのモデラーとは，旧来の CAD モデラー [48] で良く用いられているアプローチである．コマンドの蓄積で形状を表現するが，幾何的な変形操作のみしか受け付けないことと，コマンドを積む順番によって形状が大きく変化することが分かりづらいという声も多かった．そもそもが，工業製品を加工する手順を指定するシミュレーション目的の技術であったため，職人向けのモデリングシステムであった点は使い手を選んでしまうという問題があった．Wilson[49] は，形状モデリングにおけるオイラー操作を定義した．これに基づいて，Mantyla ら [50] は，オイラー操作ベースのソリッドモデラーを提案した．鳥谷ら [51] は，ソ

リッドモデリングにおけるアンドゥ・リドゥ操作を提案した．

2.2.2 ダイレクトモデラー

ダイレクトモデラーとは，ヒストリーベースのモデラーを改良し，コマンドが即形状の変形として反映されるタイプの CAD モデラーである．変形が直感的に反映できることから，近年は設計分野においても多くの場面で用いられている．しかし，目的とする形状が幾何的な工業製品向けのものである点は変わりないため，アーティスティックな造形には不向きである．造形や変形操作に対して設計意図を持たせ，拘束条件を設定することで後からの設計変更や変形操作の利便性を向上させるフィーチャーベースドモデリングの概念を用いた物が多い．例として，千賀ら [52] の手法や，Weeb[53] の手法，Tang ら [54] の手法がある．

2.2.3 スケッチインタフェース

スケッチインターフェースは，Zeleznik ら [55] によって画面上での入力を立体的なジェスチャーとして解釈する手法が提案されたものが源流となっている．五十嵐らは Teddy[56] を初めとしたスケッチインタフェースにモデリングシステムを提案している．この Teddy を発展させた 3 次元モデリングのための入力推定手法 [57] も提案している．また森ら [58] は，スケッチ入力による形状からぬいぐるみを作成する支援を行う手法を提案し，CG の利用用途を大きく広げている．

Schmidt ら [59] は立体を直接作図可能なシステムを提案した．Schmid ら [60] は，3D ペインティングシステムを提案した．Rivers ら [61] は，シルエットの入力から前後関係を解釈し，形状生成を行う手法を提案した．近藤ら [62] は，陰関数曲面を用いた集合演算によるスケッチモデリングを提案した．Gingold ら [63] は，スケッチ入力において立体構造の注釈をジェスチャーとして書き加えることで，より正確な立体形状をモデリングする手法を提案した．ANDRE ら [64] は，単視点からのスケッチによる 3 次元曲面生成を実現した．

三面図から立体形状を生成する手法として，千田 [65] の手法を初めとして，馬場ら [66] の手法，増田ら [67] の手法，八木ら [68] の手法がある．Thormählen ら [69] は，映像からキャプチャした画像を正投影することによる立体形状生成手法を提案した．

Finch ら [70] と Orzan ら [71] は，スケッチに入力に基づいた曲線に対してカラーグラデーションを付与する手法を提案した．Gingold ら [72] は，陰影の入力に基づいた曲面編集を行う手法を提案した．Wu ら [73] は，スケッチ入力から形状表面の適切な法線を復元する手法を提案した．

今泉ら [74] は，ヒートカッターを模した入力操作によって造形を行う手法を提案した．Gal ら [75] は，形状を構成する稜線を針金のように扱うことで，ユーザが直感的な変形操作を行えるシステムを提案した．

三面図から立体形状を生成する手法として，千田 [65] の手法を初めとして，馬場ら [66] の手法，増田ら [67] の手法，八木ら [68] の手法がある．Thormählen ら [69] は，映像からキャプチャした画像を正投影することによる立体形状生成手法を提案した．

2.2.4 スカルプトモデラー

スカルプトモデラーとは，形状表面に対する盛り付けや切削によって造形を行うことが可能なモデリングツールのことを指す．ユーザの入力に対して直感的な編集結果を提示することができるため，CG 制作において広く用いられるようになっている．Perry ら [76] はキャラクター造形を主目的としたスカルプトモデラーを提案している．代表的なモデラーとして Blender[77] や，Z-Brush[78]，3D-Coat[79] がある．水野 [80] はスカルプトモデリングを応用し，日本の伝統技術である「沈金」をシミュレーションする手法を提案した．利用しているデータ構造はポリゴンベースの形状表面のみを扱ったものや，ボクセルベースのソリッドモデリングが可能なものなど様々である．しかし，ユーザの入力を複雑に解釈して形状に変形を加えるため，編集手順が非可逆なものであることが，履歴を重視した運用を

行う際に問題となる．

2.2.5　シミュレーション・プロシージャルベースモデラー

近年は自然現象のシミュレーションや形状の組み合わせによってモデリングを行う研究も多くなされている．こういったプロシージャルなモデリング手法として，家具や建造物，都市景観などを表した形状を生成する研究 [81, 82, 83, 84, 85] も多数なされている．形状を構成するメッシュに対し，どのような部位を構成しているかの意味付けを持たせる手法として，Kalogerakis ら [86] の研究がある．Ovsjanikov ら [87] は，形状データを収集した Example-based のアプローチにより，特定の分野における形状の大まかな特徴を指示することで，その特徴を満たす形状を検索し，合成することでユーザの入力に適した画像を提示する手法を提案している．Chaudhuri ら [88, 89] は，収集した形状データを組み合わせてモデリングを行う際に，どの形状を接合するのがふさわしいかをベイジアンネットワークを用いて確率付けし，ユーザに提示する手法を提案している．Bronstein ら [90] は，形状データを検索するための検索ワードと形状の幾何的表現についての研究を行っている．Denning ら [91] は，ポリゴンメッシュで形状を生成する過程を示すシステムを提案した．

2.3　本研究の位置付け

これまでレビューしてきた既存研究及び技術を総括し，本研究が目指すゴールを示す．本研究の動機の発端は，人間の入力を最大限に拾い上げて活かすことのできるモデリングシステムを実現することにある．そのため，2 つのアプローチから入力方法を考えた．

1 つは人間が入力した画像からの形状生成である．第 1 章で述べたように，現状のコンピュータにおける入出力デバイスでは，射影を要する 3 次元形状を操作するよりも，2 次元のドローイングの方が直感的な編集が可能である．そこで，2 次

元画像の入力から生成した立体形状を編集することで，より簡易に3次元形状モデリングができるのではないかと考えた．画像から立体復元を行う研究は数多く存在するが，生成した形状は編集を加えるのには不向きな構造を持つことが多く，望ましい復元結果が得られなかった場合はその手法による復元を断念せざるを得ない．そこで，本研究では編集形状を点群で表現することにした．点群を用いることで，位相を気にせずに形状編集ができるようになる．また，個々の点に画像から抽出した色を反映することで，画像の内容に沿ったスカルプトが可能になる．

もう1つはヒストリーを活かしたモデリングシステムの考案である．ヒストリーは，人間がモデリングシステムに対して行った操作の完全な履歴であり，これがあればアンドゥ・リドゥが自由にできるのはもちろんのこと，他人が行った造形操作を共有し，手順を自由に組み替えて新たな形状を作り出すことも可能になるはずである．このようなアプローチは，CADのヒストリーベースモデラーや，ハーフエッジ構造を用いたモデリングツールにおいては実現可能である．しかし，変形コマンドや頂点の移動，分割処理などはデータ構造を把握していなければ行えない操作である．これでは人間にとって自然な形状入力手段とは言い難い．現状のモデリングツールにおいては，スケッチのような輪郭線入力や，各種スカルプトモデラが実現しているような切削・盛り付け入力が，人間にとっての自然な形状入力手段として提供されている．しかし，このような入力はあくまで形状データを編集する上でのトリガーでしかなく，コンピュータ上で取り扱うのはそのような入力が蓄積されたことによる結果としての形状のみである．このような状況では，ユーザの入力を元に可逆的な操作を定義できないことが多く，履歴に基づいた操作を実現することが困難である．

そこで，本研究では「ストロークベース」という新たなモデリングメソッドを提案する．これはスケッチベースとは異なり，ユーザが画面上で入力した軌跡をその都度保持し，その軌跡を3次元空間に投影することで形状を編集していくアプローチである．ユーザが入力した情報を完全に保持することで，履歴を自由に操作することが可能になる．また，ストロークを射影して3次元形状を生成する

にあたって，陰関数集合に変換することで，スカルプト操作を CSG モデルに基づいたブーリアン操作に集約することができた．これにより，履歴を単純に行き来するだけではなく，他のユーザが入力したストロークと自分の作成したデータを融合したり，特定の手順を反復したりといった柔軟な運用が可能になる．

これらのアプローチにより，データ構造に偏りがちであった 3 次元形状モデリングにおいて，ユーザの入力を重要視した手法を構築することができるようになった．ユーザの入力を保持することにより，単純な編集の利便性向上に留まらず，プロセスベースによる新たなモデリング手法を確立した．

第 3 章

点群とボクセルを併用した
形状試作モデリングシステム

3.1 はじめに

本章ではラスター画像によるイラストをベースとした，新しい形状モデリングシステムを提案する．この提案手法においては，立体形状のデザイン工程において初期段階に描かれるラフスケッチから3次元モデルを再構成する．このスケッチには厳密な立体的整合性を考慮する必要がない点が，これまでの手法と比べた場合のメリットである．そしてシステムのユーザは，生成した形状に対して切削や盛り付けなどの編集操作を行うことができる．このシステムによって，スケッチ画像から大雑把な立体を作成し，手軽に修正を施すことで立体形状の簡易的な試作が可能になる．

本章にて提案するシステムは，形状を点群サーフェスとボリュームデータによって保持する点がもう一つの特徴である．このデータ構造は，画像のシルエットによる初期形状の生成と，ユーザの直感的な操作による形状編集を実現する．ポイントセットによる形状表現は，ポリゴンのように点を結んで面を張る必要がない．このため，微細な特徴を点の追加だけで表現することができる．点の集合のみでは形状の内外情報を保持することが難しいため，ボリュームデータを併用することにした．ボリュームデータに対して画像のシルエットによる交差判定を行うことにより，複数のシルエットに基づいた内外情報を初期値として持つことができる．内外情報を厳密に保持できる形状データ構造としては，陰関数曲面やブール集合による表現形式も存在する．これらの形式の場合，複雑な編集をしていくと関数や基本形状の情報が加算されていくため，ユーザの操作による直感的な編集が困難になる．ポイントセットとボリュームデータによる形状表現の場合は常に編集結果を保持するのみであり，常にユーザの操作は直接反映できる．

3DCG における形状モデリングには様々なアプローチが存在する．モデリング手法は大きく分けるとスキャナなどによるサンプリングベースの手法と，ユーザの操作による手続き的な手法に分けられる．正確な3次元形状の構築には高度な空間認識能力が必要となるため，ユーザによる手続き的なモデリングをサポート

するシステムやインタフェースが数多く提案されている．

CAD の分野においては，対象とする物体を正面・側面・上面から描いた 3 面図を用いた形状設計を行うことが多い．3 面図から立体形状を生成する研究も古くからなされているが [65]，入力できるのは正確に平行投影された図面に限られる．デザインの初期段階においては，デザイナーは形状を試行錯誤しながら描くため，常に立体的整合性を意識しながらデザインすることは困難である．Schmidt [59] らは立体を直接作図可能なシステムを提案したが，ベクトルベースの入力ではデザイン初期段階のラフデザインを取り入れることが難しい．

このような制限があるため，厳密な投影図を用意できない場合は手動によるモデリングシステムを採用することになる．自動生成によらない手続き的なモデリングにおいても，事前に 2 次元画像上で形状をスケッチするのが一般的である．モデリングシステム上においてはスケッチした画像は参照用の素材として扱われ，モデリングの作業自体には直接的な役割を与えないことが多かった．そこで，ユーザのドローイングプロセスを立体形状生成に取り入れる研究が行われるようになった．なかでもユーザの手書き入力から立体形状を推定して生成する研究は，Sketch [55] や Teddy [56] を始めとして数多くの提案がなされてきた．

Sketch [55] や Teddy [56] のインタフェースはユーザの 2 次元ドローイングプロセスを利用した立体生成が目的であり，既存の画像から立体を生成するわけではない．既存の画像をトレースして立体生成を行うことは可能だが [63]，そのトレース作業とポリゴン化処理において画像の微細な特徴やタッチは損なわれてしまう．

画像から直接立体形状を復元する研究も数多く存在する [68] が，全自動で完璧な立体復元は困難である．イメージベースドモデリングにおいては，入力画像には厳密なカメラパラメータを設定する必要があるため，カメラパラメータが未知な画像は入力することが不可能か，あるいはコンピュータビジョン技術 [92] によってパラメータを逆算する必要があり，ドローイング画像を適用することは困難である．また，これらのプロセスによって復元された形状は一般的に修正や編集作業に不向きな構造になることが多く，意図しない形状に対する修正操作や調整が

難しい．

本研究で対象とするのは，必ずしも正確とは限らない投影画像，あるいは投影を想定して描かれた画像を入力とするケースである．人間の手によってドローイングされた画像をモデリングのベースとして最大限に活用し，かつ細かな補正や編集作業を効率的に行う環境が存在すれば，元々立体化を想定していなかった画像の立体化や，ラフなスケッチをベースとした立体形状のモックアップ，あるいはプロトタイプ生成に有用であると考えられる．

同じような研究として，Kara らの研究ではスケッチ入力に基づいた電気回路図の自動構成 [93] や，スケッチ入力のラインによってテンプレート形状を意図通りに変形させて形状編集を行うシステム [94] が提案されている．このように，既存の形状を元にスケッチインターフェースによって新しいデザインを作る研究があるのは，初期のデザインにおける，スケッチインターフェースの需要がある顕れである．従って，三面図を元に生成した初期形状から，スケッチインターフェースによって形状を編集する本システムも，初期デザインプロセスの効率化に大きく貢献できる．

そこで本研究では，以下の 2 点を備えたシステムを提案する．

1. 入力画像に厳密な立体的整合性を求めず，画像のシルエットを基にした大雑把な初期形状を生成する．

2. 生成した初期形状はボクセルデータとサーフェスデータの二重構造によって構成し，切削や盛りつけなどの直感的な編集操作が可能なものとする．

本章では以上 2 点を満たしたモデリングシステムを提案し，実装することで有用性を検証した．

3.2 本システムの概要

本システムにおける形状作成手順は以下の通りである．各手順において用いるアルゴリズム，データ構造については後の節 3.3, 3.4 と 3.5 で述べる．以下の図

3.1 は本システムのフローチャートを示す.

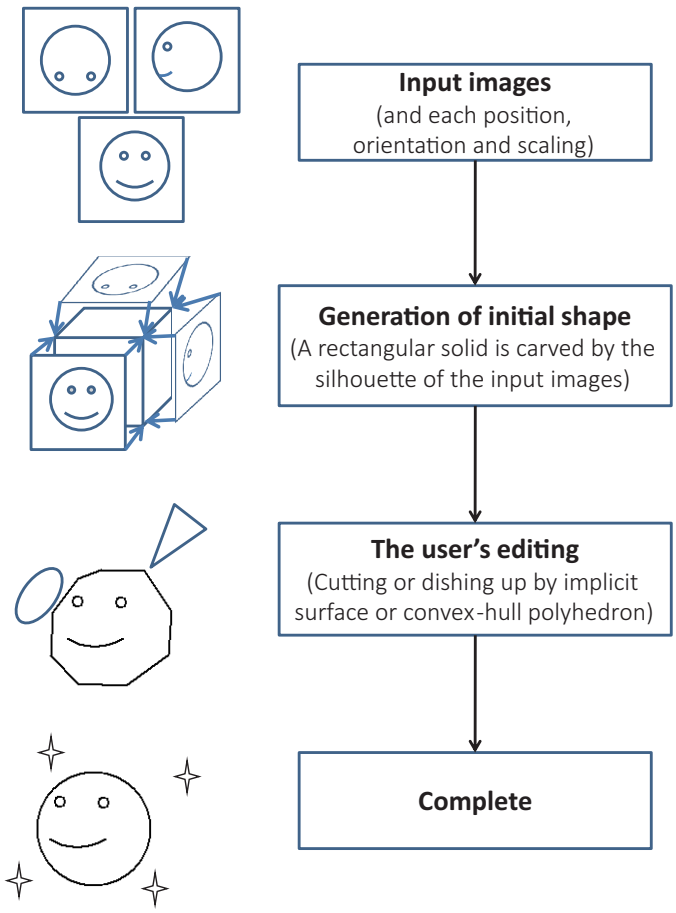


図 3.1: 本システムのフローチャート

システムのユーザは，作成しようとしている形状を描いたスケッチ画像と，それぞれの画像について空間中の位置，回転姿勢，スケールを入力する．これは画像の座標系からモデリング形状への座標変換を行うための情報である．位置は 3 次元ベクトルで定義し，回転の姿勢は 2 つの直交するベクトル (画像の幅方向ベクトルと縦方向ベクトル) によって与える．画像 I を座標 $\mathbf{o}$ に配置し，幅方向ベクトルを $\mathbf{l}$，縦方向ベクトルを $\mathbf{m}$ とし，スケールを s とする．この時の画像上の 2 次

元座標 (p'_x, p'_y) を 3 次元座標 $\mathbf{p}$ に変換する計算は，次の式 (3.1) によって行う．

$$\mathbf{p} = \mathbf{o} + sp'_x\mathbf{l} + sp'_y\mathbf{m} \tag{3.1}$$

画像から立体復元を行う場合，基本的には 3 面図を想定した画像を入力するが，本システムでは任意の視点から描いた画像も入力可能である．画像の枚数は 3 枚に満たなくても構わないが，初期形状の粗さが際だってしまい，ユーザの手作業による修正の必要性が増加してしまう．このため，形状のシルエットが変化して見えている 3 枚以上の画像入力が望ましい．次の図 3.2, 3.3, 3.4 に，本システムで想定している入力画像を示す．

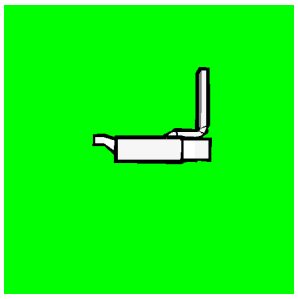


図 3.2: 正面図の入力画像

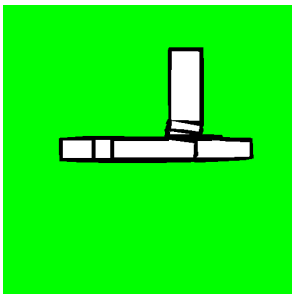


図 3.3: 側面図の入力画像

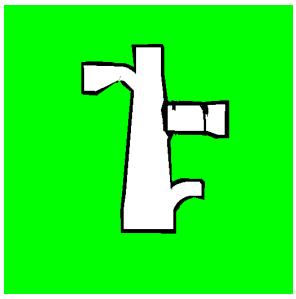


図 3.4: 上面図の入力画像

次にシステム側で入力画像を元に視体積交差法 [32, 33] を適用し，大雑把な初期形状を生成する．視体積交差法を適用する対象は入力画像をテクスチャマッピングした直方体状とする．入力画像には透明度を用いて形状の内外を表すシルエットの情報を設定しておき，この透明度の情報に基づいてくりぬき操作を行う．くりぬいた形状の表面にも入力画像を投影することでテクスチャマッピングを行い，画像の特徴を直接形状に反映させている．次の図 3.5, 3.6 に，本システムで生成する初期形状を示す．

生成した形状に対して，ユーザが切削・盛りつけなどの編集操作を行う．編集操作は曲線的な形状を表す陰関数曲面と，直線的な形状を表す凸包ポリゴンを用いることができ，画面上の形状に対してスイープ操作を行うことで切削，または盛り付けを行う．一般的なスカルプトモデリングとは異なり，形状の表面をポイン

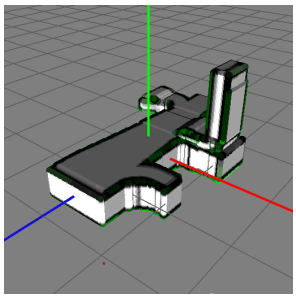


図 3.5: 生成した初期形状 (1)

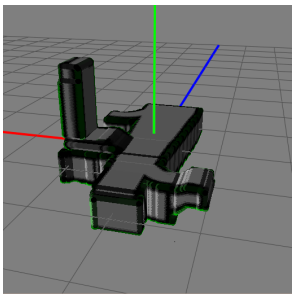


図 3.6: 生成した初期形状 (2)

トセットによって表現しているため，ユーザの細かい操作のニュアンスをそのまま反映させた形状編集が可能である．切削後のポイントセットにも入力画像に基づいた色付けを行っており，入力画像の特徴を活かしたままの編集を進めることができる．次の図 3.7, 3.8 に，本システムで提案する切削操作のイメージを示す．

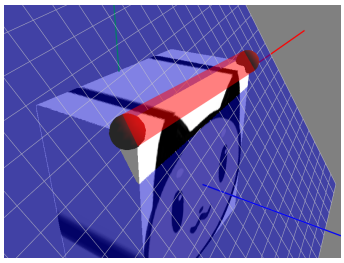


図 3.7: 切削操作入力のイメージ

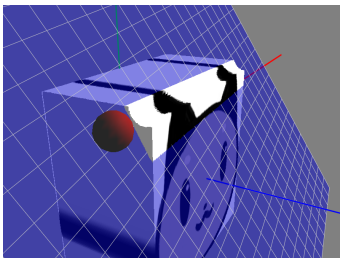


図 3.8: 切削操作適用後のイメージ

3.3 二重データ構造

本手法では，モデリングの対象とする形状データを，形状の内外情報を持つボリュームデータと，形状の表面を表す点群データの二重構造によって構成する．ボリュームデータとは別個に点群サーフェスデータを保持することによって，点群データはボクセル解像度の制約を受けずにサーフェスを表現することができる．このデータ構造によって，画像からの形状生成とユーザによる直感的な編集操作を実現した．

点群データは，形状の表面を表す頂点の位置ベクトルと，その頂点における法線ベクトルの配列として定義する．本章では，この位置ベクトルと法線ベクトルの配列を点群サーフェスデータと呼称する．このアプローチは Pfister らの手法 [43] を踏襲したものである．

ボリュームデータは形状内 (1) と形状外 (0) の 2 値を持つ．形状を編集する度に，ボリュームデータ側の内外情報も合わせて更新する．点群データは初期形状の生成時とユーザの形状編集によって生成・更新する．形状を削る操作を行った場合，まずボリュームデータが更新され，その内外情報と切削を行う断面を表す形状に基づいて点群サーフェスを生成する．切削の断面を表す形状には陰関数や凸包による表現を用いる．点群サーフェスが保持する頂点数はボクセルの解像度によらず，編集を行う際に入力する切削領域のサンプリング精度によって自由に増減することができるため，ボクセルの格子間隔とは関係なく頂点を生成することができる．

3.4 画像のシルエットに基づく形状生成

本システムにおける編集形状は直方体からスタートし，画像のシルエットを用いたくり抜き操作を行い，画像の特徴をおおまかに反映した初期形状を生成する．編集形状は直方体状の時点からボクセルと点群データによって構成している．形状内部に位置するボクセルには初期値として 1 を，形状外部には 0 をセットする．ボクセルで表している領域の外側は形状の外部として扱う．本システムはこのデータ構造から，入力イラストをベースとした初期形状を生成する．この初期形状はユーザが希望する形状を作成するためのガイドラインとして機能する．

システムは入力画像のシルエットに基づいて直方体を削っていく．画像のシルエットは各ピクセルに割り当てられた透明度に基づいて決定する．したがって，入力画像にはシルエットを表現した透明度を各ピクセルに割り当てる必要がある．本システムではボクセル値と同様に，形状内部を表すピクセルには不透明状態を表す値 (1) を，形状外部を表すピクセルには透明状態を表す値 (0) を用いた．この入

力画像を，ボクセルで表現した直方体形状に対して射影することで，各画素がどのボクセルに対して直交するかを求める．各画素とボクセルの対応関係は，ボクセルの座標値から次の式 (3.2) によって求めることができる．

$$\begin{pmatrix} p'_x \\ p'_y \\ p'_z \end{pmatrix} = \mathbf{T}^{-1}(\mathbf{p} - \mathbf{o}) \tag{3.2}$$

(p'_x, p'_y) は入力画像上の 2 次元座標である． 3 次元座標 $\mathbf{p}$ はボクセルの中心位置を表す．3 次元空間中における入力画像の位置と姿勢は，3.2 で定義した位置座標 $\mathbf{o}(o_x, o_y, o_z)$，画像の幅方向ベクトル $\mathbf{l}(l_x, l_y, l_z)$ と，画像の高さ方向ベクトル $\mathbf{m}(m_x, m_y, m_z)$ によって表すので，これらのパラメータを用いて画像座標を 3 次元空間座標に変換する 3×3 の行列 $\mathbf{T}$ を次式 (3.3) のようにして求める．

$$\mathbf{T} = \begin{pmatrix} l_x & m_x & g_x \\ l_y & m_y & g_y \\ l_z & m_z & g_z \end{pmatrix} \tag{3.3}$$

ここでの $\mathbf{g}(g_x, g_y, g_z)$ は，$\mathbf{l}$ と $\mathbf{m}$ の外積ベクトルを表す．この計算の結果，求めた (p'_x, p'_y) が画像の領域外を指す値になる場合があるが，この時はその画素は形状外部を表す透明な画素として扱う．

入力画像を任意の平面から投影することにより，1 つの画素は直方体を構成する複数のボクセルと交差することになる．システムは交差したそれぞれのボクセルに対して，画素が持つ透明度が 0 の場合にボクセルの値を 0 に設定し，形状外部であるとして記録する．言い換えれば，直方体を構成する全てのボクセルは，射影された画素の値が透明である場合に削り取られるということである．

複数の正投影画像を入力とした場合，システムは全ての画像の画素と個々のボクセルが交差しているかを判定する．交差する画素のうち，1 つでも透明な画素が見つかった場合はそのボクセルの値を 0 にして，形状外部としてマークする．この工程は，彫刻における初期段階の作業を模したものであり，木材の塊から粗い形状を削り出した段階に相当する．

入力画像が持つ透明度を用いて形状の外部となっているボクセルをマークしたのち，システムは形状の内外の境界となっているボクセルの境界頂点に対して点

群によるサーフェスを生成する．法線ベクトルは隣接ボクセルの接平面を抽出することで，個々の点に対して設定する．

生成したそれぞれの点の色は，入力画像によるテクスチャマッピングによって求める．複数の画像を入力した場合はそれぞれの画像の色に重み付けをしたブレンドによって計算する．画像ごとの重みは点が持つ法線ベクトルの向きと，画像に設定済みの姿勢によって決定する．ここで画像を k 枚入力するとし，それぞれに設定している位置と姿勢を $\mathbf{o}_i, \mathbf{l}_i, \mathbf{m}_i$ とする．色を決定しようとしている点 p が持つ法線ベクトルを $\mathbf{n}_p$ とし，各画像において空間座標 $\mathbf{p}$ を画像上の座標に変換した時の座標を $\mathbf{p}'$，その画素が持つ色値を $\mathbf{I}(p'_x, p'_y)$ としたとき，点 P の色値 $\mathbf{c}_p$ は次の式 (3.4) によって求める．

$$\mathbf{c}_p = \sum_{i=1}^{k} \big(|\mathbf{n}_p \cdot (\mathbf{l}_i \times \mathbf{m}_i)| \big) \mathbf{I}(p'_x, p'_y) \tag{3.4}$$

三面図を入力した場合，各画像に設定される姿勢ベクトルは一定の値を持つ．それぞれの画像において $\mathbf{l}_i \times \mathbf{m}_i$ が示すベクトルは直交するため，全ての点において各画像に対する重みの総和は 1 となる．すでに入力している画像の反対方向から描いた画像を入力する場合は，式中の絶対値を取る部分を $\max(\mathbf{n}_p \cdot (\mathbf{l}_i \times \mathbf{m}_i), 0)$ によって置き換えて，負の値をクランプする．これによって双方向から描いた画像を反映させることができる．任意視点からの画像を追加する場合は，各画像に対する重みの総和が 1 を超えるケースが想定される．本章における実装では任意視点からの画像を点の色に反映させることは想定していないが，正確に反映させる場合は各画像間の姿勢ベクトルの角度を算出し，重みを調整する必要がある．

3.5 形状編集

本節では入力画像のシルエットから生成した初期形状に対して，システムのユーザーが修正を施す際の編集操作の詳細について述べる．本システムではユーザーに対して，切削と盛り付けの 2 種類からなる編集操作を提供する．それぞれの編集操作における操作領域は，陰関数表現による曲面か，凸包形状によって定義し

た形状を，用途に合わせて用いることができる．それぞれの操作においては，点群による表面形状と，ボクセルによる内外情報を同期的に更新していく．これらの詳細を次項以降にて述べる．

3.5.1 編集位置の計算

ユーザーは編集箇所をスクリーン上からポインティングデバイスによって入力する．スクリーン上の座標値は2次元であるため，そのままでは3次元空間内の座標を一意に決定することができない．本システムでは仮想の平面を画面内に用意し，デバイスの入力によって得たスクリーン座標を仮想平面上に射影する．OpenGL [95] で使用しているモデルビュープロジェクション行列を用いて，スクリーン座標を3次元空間中に存在するカメラの位置を基準に逆変換し，その座標を端点としたカメラの視線方向と平行な半直線を求める．この半直線が仮想平面と交差した点を，ユーザが入力した編集箇所として扱うことで，編集領域の中心となる位置座標を一意に決定する．この仮想平面はマウスドラッグによって自由に回転し，移動することができる．ユーザは視点と仮想平面の位置及び姿勢を制御することで，意図したとおりの編集を行うことができる．

スクリーン座標をモデル座標系に変換する処理は，次のようにして行う．(x, y) をポインティングデバイスによって入力したスクリーン座標とする．これに仮想的な深度値を与えて $\mathbf{d}(x, y, 1.0)$ を定義する．$\mathbf{d}$ はビュー座標系におけるベクトルであり，任意の正の値 t とのスカラ積 $t\mathbf{d}$ は，スクリーン上の座標 (x, y) が指し示しうる座標を表すことになる．このベクトル $\mathbf{d}$ は次の式 (3.5) によってモデル座標系へ変換できる．

$$\mathbf{d}' = (\mathbf{PM})^{-1}\mathbf{d} \tag{3.5}$$

$\mathbf{P}$ はプロジェクション行列を，$\mathbf{M}$ はモデル行列を表す． この $\mathbf{d}'$ を用いて，スクリーン座標から変換した直線上の点 $\mathbf{p}$ は，次の式 (3.6) によって求めることができる．

$$\mathbf{p} = \mathbf{e} + t\mathbf{d}' \tag{3.6}$$

$\mathbf{e}$ はカメラの位置座標を表す．

編集位置決定用の仮想平面は，平面上のある 1 点と，平面に対して垂直な法線ベクトルによって定義できる．平面上のある 1 点の座標を $\mathbf{a}$ とし，法線ベクトルを $\mathbf{b}$ とした時，$\mathbf{a}$ とは異なるもう 1 点の座標 $\mathbf{p}$ と次の関係式 (3.7) が成り立つ．

$$(\mathbf{a} - \mathbf{p}) \cdot \mathbf{b} = 0 \tag{3.7}$$

編集位置の座標 $\mathbf{q}$ は編集平面と，スクリーン座標から変換した直線が交差する点によって求められるので，式 (3.7) と 式 (3.8) によって計算できる．

$$\begin{aligned} (\mathbf{a} - \mathbf{q}) \cdot \mathbf{b} &= 0 \\ \mathbf{q} &= \mathbf{e} + t\mathbf{d} \end{aligned} \tag{3.8}$$

ツールの位置 $\mathbf{q}$ は，陰関数曲面や凸包を切削領域として定義する際の基準点として用いる．編集形状をどのくらいの深さまで切削するかを制御する際は，編集平面を法線方向 $\mathbf{d}$ に移動することによって対応する．奥行きを指定することによって，ユーザは変形の度合いをコントロールすることができる．

3.5.2 陰関数曲面による切削操作

ユーザーは陰関数表現した曲面を用いることで，滑らかな曲面によって切削を行うことができる．本システムで用いる陰関数曲面は，式 (3.9) によって定義できるものである．

$$f(x - q_x, y - q_y, z - q_z) = 0 \tag{3.9}$$

そして，この曲面によって内包する空間は式 (3.10) によって表す．

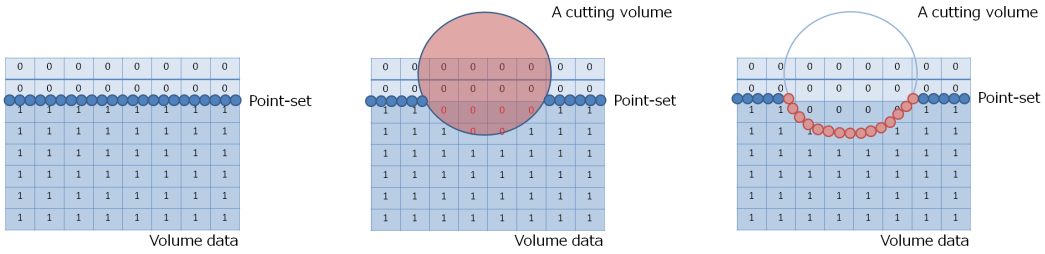
$$f(x - q_x, y - q_y, z - q_z) < 0 \tag{3.10}$$

この時の (q_x, q_y, q_z) は，式 (3.7), (3.8) によって求めたユーザーの入力による編集位置座標である．例えば，陰関数曲面を半径 r の球体として表す場合，陰関数 f は式 (3.11) によって定義できる．

$$f(x, y, z) = x^2 + y^2 + z^2 - r^2 \tag{3.11}$$

切削操作は，切削による形状の状態更新と，切削後の断面生成の二段階によって完了する．まず形状の内外判定を保持するボリュームデータに対し，領域内に含まれているボクセルの値を，形状外 (0) として更新する．次に点群サーフェスデータに対して検索を行い，切削領域内に含まれている点群を全て消去する．次は断面となる点群の生成を行う．まず，切削領域である陰関数曲面のサンプリングを行い，陰関数の値が 0 となる等値面上の位置ベクトル α を抽出する．この等値面上の点 α を含むボクセルに対して，その周囲に形状内の状態値 (1) と形状外 (0) の状態値を持つボクセルが両方存在する場合，点 α を含むボクセルは切削を行って生じる断面を含むボクセルであると言えるので，点群サーフェスデータに対して位置ベクトル α を切削断面上の新たな点群として追加する．

位置ベクトル α の追加と同時に，法線ベクトルの算出も行う．断面上の点 α の法線ベクトル $\mathbf{n}_\alpha$ は，切削領域が球状に定義されていることから，切削領域の中心位置ベクトル $\mathbf{q}$ を用いて $\mathbf{n}_\alpha = \mathbf{q} - \alpha$ と表すことができる．これにより，切削操作を行うごとに適切な法線を各頂点に与えて，形状に忠実なライティング処理を行うことができる．以上の処理を，切削領域の等値面上に存在する全ての点に対して行う．図 3.9 の (a)(b)(c) は，切削操作時におけるデータの変化を示したものである．



(a) 切削を行う前の状態　(b) 切削途中段階の状態　(c) 切削完了後の状態

図 3.9: 切削操作中のデータの変化

図 3.10 に示す画像は，切削操作の始点と終点を球体で示し，その間をスウィープした領域を切削操作の対象領域として示している．実際に領域内の切削を行った結果を示したものが図 3.11 である．

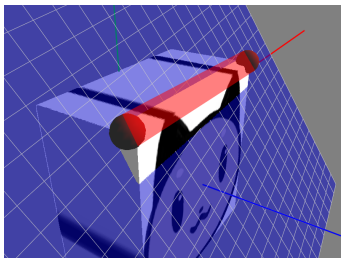


図 3.10: 陰関数による切削操作入力

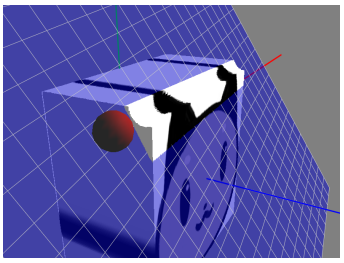


図 3.11: 陰関数による切削操作結果

3.5.3 凸包形状による切削

シャープな形状を造形したい場合，ユーザーは三角形の集合による凸包形状によって切削操作を行うことができる．凸包の条件を満たした形状であれば，ユーザーは用途に合わせて異なる形状を操作領域として入力することができる．

凸包形状に対する任意の点の内外判定は式 (3.12) で行う．k 枚の三角形ポリゴンで構成する凸包形状 (ただし $k \leq 4$) を想定し，$\mathbf{h}_{i,1}$, $\mathbf{h}_{i,2}$, $\mathbf{h}_{i,3}$ は凸包を構成する三角形の頂点を表す．i は三角形の ID ($1 \leq i \leq k$) を表す．$\mathbf{n}_i$ は，i 番目の三角形における法線ベクトルを示す．これらを用いて表現した凸包を切削領域とした場合の切削対象となる頂点集合を S とする．

$$\begin{aligned} S_i &= \{\mathbf{p} | (\mathbf{p} - \mathbf{h}_{i,1}) \cdot \mathbf{n}_i < 0\} \\ S &= S_1 \cap S_2 \cap \cdots \cap S_{k-1} \cap S_k \end{aligned} \tag{3.12}$$

β は，凸包形状の表面上における任意の点をサンプリングした集合とする．i 番目の三角形上における任意の点は，式 (3.13) によって表す．

$$\beta_{i,u,v} = \mathbf{h}_{i,1} + u(\mathbf{h}_{i,2} - \mathbf{h}_{i,1}) + v(\mathbf{h}_{i,3} - \mathbf{h}_{i,1}) \tag{3.13}$$

ここで，$u = \Delta a$ (a は $0 < a < 1$ を満たす実数)，$v = \Delta b$ (b は $0 < b < 1$ を満たす実数) とする．サンプリングの間隔は必要な解像度に応じて変更する必要がある．法線ベクトル $\mathbf{n}_{i,u,v}$ は，サンプリングした点 $\beta_{i,u,v}$ の i に対応した，i 番目の三角形の法線ベクトルを利用して，式 (3.14) の通りに求める．

$$\mathbf{n}_{i,u,v} = -\mathbf{n}_i \tag{3.14}$$

新たに生成する点群サーフェスは，$\beta_{i,u,v}$ で表現できる点のうち，対応するボクセルの状態値と周囲のボクセル値を参照して，内外の境界となっていることが判明した点を該当する点として生成する．このように生成した点群サーフェスは，切削によって生じた形状の穴をふさぐように配置する．

図 3.12, 3.13, 3.14, 3.15 は，凸包形状による切削操作の様子を示している．

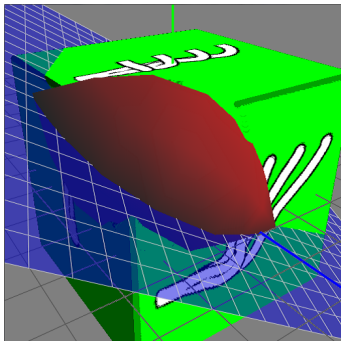


図 3.12: 凸包による切削操作入力 (1)

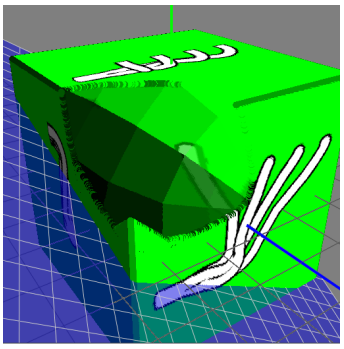


図 3.13: 凸包による切削操作結果 (1)

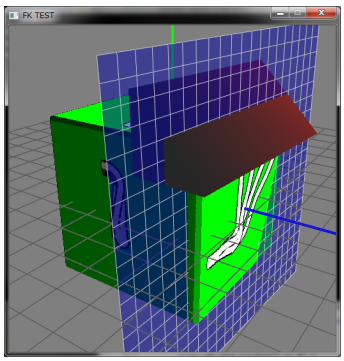


図 3.14: 凸包による切削操作入力 (2)

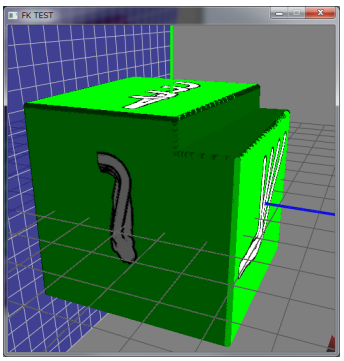


図 3.15: 凸包による切削操作結果 (2)

3.5.4 盛り付け処理

盛り付け処理は，切削操作と真逆の操作を行うことで実現可能である．すなわち，次の手順になる．

1. 切削領域内の点群を削除する．

2. 切削領域の表面をサンプリングし，近傍に形状外部を示す値 (0) のボクセルがあれば断面となる点群サーフェスを生成する．

図 3.16,3.17 に，切削を行った部位に対する盛り付けを行っている様子を示す．

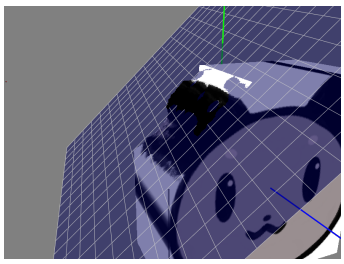


図 3.16: 盛り付け操作の入力

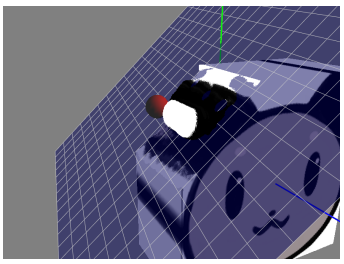


図 3.17: 盛り付け操作の結果

3.6 実装と検証

ここまでで述べた手法に基づき，PC 上での実装実験を行った．今回は評価実験のため，入力画像には既存の CAD データをイラスト画調にレンダリングしたものを用いた．開発言語には C++を用いて，グラフィクスシステムとしては OpenGL をベースとした FK Toolkit system [96] を用い，一部処理には NVIDIA 社の Cg 言語と CUDA を用いた．次の表 3.1 に，今回の実験で使用した環境を示す．

表 3.1: 検証を行ったコンピュータの動作環境

CPU	Intel Core 2 Duo E8400 @3GHz
RAM	4GB
GPU	NVIDIA GeForce 9800GTX+

今回の実装ではボクセルの解像度を $256 \times 256 \times 256$ とした．正面図，側面図，上面図を想定した画像を入力して初期形状を入力したところ，元の形状の特徴を持った初期形状が生成できた．しかし，小さな凸部が大きな凸部の影に隠れている場合，あるいは凹面を持った形状の場合は，その特徴がシルエットに現れない．これらの場合は初期形状に特徴を正確に反映させることができないが，切削によ

る編集で容易に修正を施すことが可能であった．初期形状で生成した点群には入力画像によって色付けしてあるため，直感的な編集環境を実現できた．

図 3.18 は入力画像の例であり，図 3.19, 3.20 は初期形状の生成結果である．今回のテストで生成した形状は，初期状態では 40000 頂点によって構成され，編集を経て 300000 頂点になった．この形状は画像では表現しきれない凸部を含んでいたため，意図した形状を再現するために若干の切削操作を必要とした．その切削操作の前の様子を図 3.21 に，切削操作後の様子を図 3.22 に示す．形状は画像の特徴をおおまかに反映しており，プロトタイプとして十分な品質を持っている．

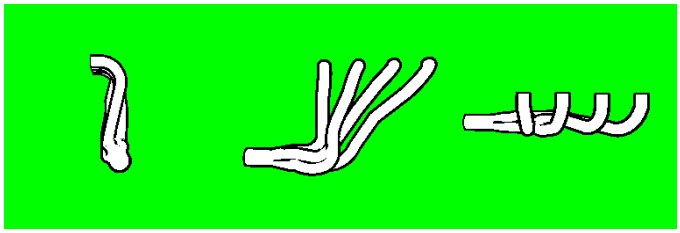


図 3.18: 排気管を描いた入力画像

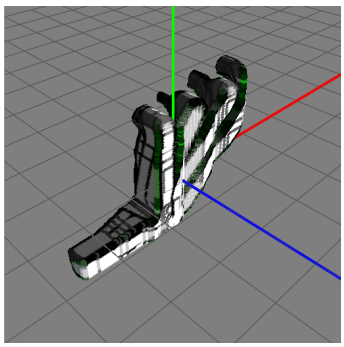


図 3.19: 排気管の初期形状 (1)

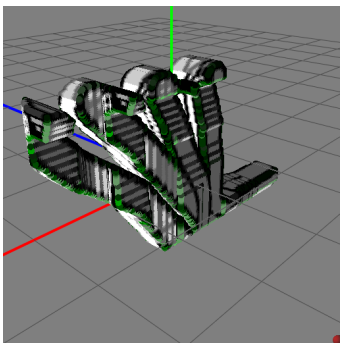


図 3.20: 排気管の初期形状 (2)

図 3.23, 3.24, 3.25 は三面図の入力画像の例であり，図 3.26, 3.27 はシステムによって生成した初期形状である． 今回のテストで生成した形状は，初期状態では 34000 頂点によって構成され，編集を経て 110000 頂点になった．この形状も画像の特徴をおおまかに反映しており，プロトタイプとして十分な品質を持っている．この形状における編集操作は，一部に余白である緑色の部分が残っていたた

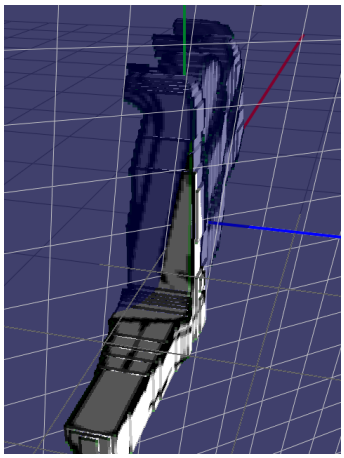


図 3.21: 編集前の排気管形状

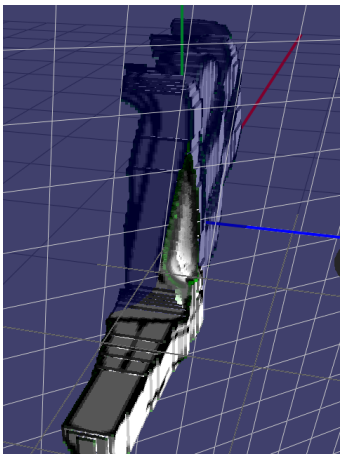


図 3.22: 編集後の排気管形状

め，これを削り取る作業を行う程度で十分であった．切削操作の前の様子を図 3.28 に，切削操作後の様子を図 3.29 に示す．

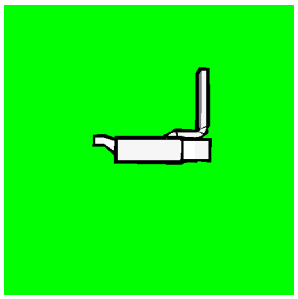


図 3.23: 正面図

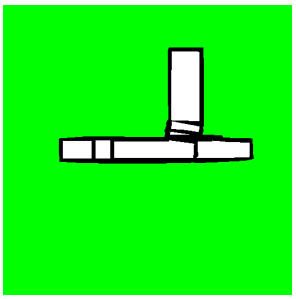


図 3.24: 側面図

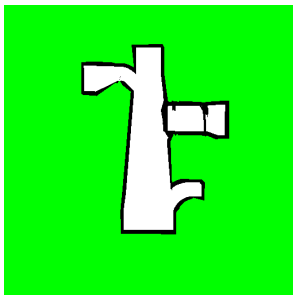


図 3.25: 上面図

3.7　考察

3.7.1　点群による形状表現

ボクセルのみで形状を表現した場合，表現可能な形状の細かさはボクセルの解像度に依存する．単純なボクセルデータをそのまま用いると，解像度の増加に際してデータ容量が指数関数的に増加してしまう．n をボクセルの 1 軸あたりの解像度とした場合，ボクセルデータは 3 軸存在するため，$O(n^3)$ のオーダーで容量が増加することになる．このようなデータ容量の増加を防ぐため，効率的なデータ構

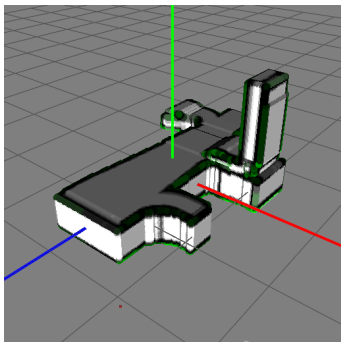


図 3.26: 三面図から生成した形状 (1)

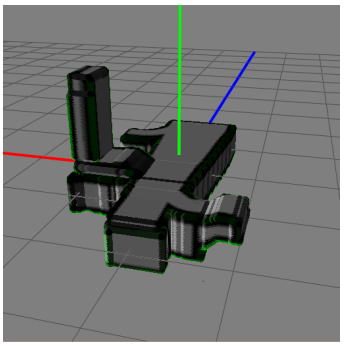


図 3.27: 三面図から生成した形状 (2)

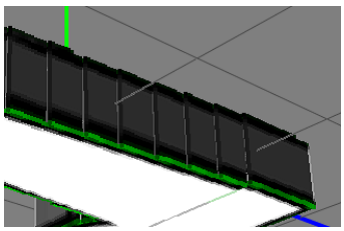


図 3.28: 編集前の形状

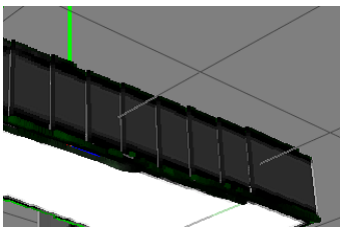


図 3.29: 編集後の形状

造としてオクトツリーが存在する．オクトツリーは，1 つのセルを 8 つに分割するツリー構造を持ち，部分的に高解像度のデータを表現できる．しかし，それでも詳細な編集によるデータの肥大化は避けられない．形状表面を点群として別データとして保持することによって，切削や盛り付け操作の際に領域表面をどの程度の解像度でサンプリングするかを制御することで，ボクセルの解像度を変更せずとも詳細な表面形状を保持することができる．表面を表す頂点の増加は，データ構造の変更が必要な手法と比較しても，データサイズや参照速度の面で有利である．

3.7.2 処理速度の検証

本システムをクロック数 3.0GHz の 2 コア CPU 上で稼働した際の切削・盛り付け操作において，256 × 256 × 256 のボクセル，300000 個の点群を対象とした場合の処理時間は 10(ms) であった．これはインタラクティブなモデリングを行うのに十分な処理速度だと言える．本実装においては，点群の各頂点がどのボクセル空

間に属しているか分類することにより，点群の増減処理における探索時間を大幅に削減した．点群の分類は多数の頂点を持つ形状編集において有効な手法である．

3.8 まとめ

本章では，機械部品のような形状を 2D のイラスト画像から直接彫り出すシステムを提案した．このシステムは複数の画像に基づいて簡易的な初期形状を生成するが，その画像には厳密な立体的整合性が不要である．ポイントベースのサーフェス表現を用いたことで，システムがユーザの希望をある程度満たすような初期形状を生成し，それに対してユーザが切削を主とした直感的な編集が可能な手法を提案した．点群を直接形状として取り扱うことにより，点単位での色付けと面の制約を受けることのない自由な形状操作が可能となり，3 次元形状モデリングに新たなアプローチを示すことができた．

本手法によって生成した点群形状は，そのままでも表示に耐えうるものであるが，既存のポイントグラフィクスのレンダリング技術や，ポリゴンへの再構成処理などによって，様々な用途への応用が期待できる．形状設計の場面においても，ラフなスケッチからの簡易的な立体化手法を応用することで，モックアップ製作やプロトタイピングが容易になることが期待できる．

今後の展望としては，感圧式のペンタブレットなどのデバイスとの連動による精細な切削操作や，編集を繰り返すことで不均一になっていく点群の再配置，整列処理などを実現することで，より有用な形状モデリングシステムへと発展することが期待できる．

第 4 章

陰関数集合によるストロークベースのモデリングシステム

4.1 はじめに

近年 3DCG のモデリング手法として，スカルプトモデリング [76, 77] が注目されている．スカルプトモデリングとは，粘土に対する彫刻を行う感覚で立体造形が可能なシステムであり，主にペンタブレットを用いて操作する．従来のポリゴンやサーフェスの操作を主体とするモデラーとは異なり，ボクセルやポリゴンメッシュ上における変位マップを操作することで造形を行うため，アーティストの感性を直感的に反映させやすい．そのため，近年の CG を用いた映像コンテンツやゲーム作品において採用事例が多く見られるようになった．スカルプトモデリングの主目的は，形状表面の凹凸を操作することによって，仮想粘土の切削や盛り付けといったメタファによる操作を実現することにある．代表的なスカルプトモデラーである Pixologic 社の Z-Bursh [78] では，ペンタブレットからの入力に従ってポリゴンを再分割して変位を与えることで，形状の編集を行っている．スカルプトによって生成した高ポリゴンモデルから低ポリゴンに対する変位マップを作成する [6] のが一般的な利用方法であるが，この場合は形状の位相が大きく変化する造形を行うことはできない．一方でボクセルをベースとしたスカルプト操作を実現する手法 [28, 30, 31] も提案されており，幾何変形のみに留まらない位相変形を伴う造形操作も可能としている．モデラーソフトの中では PILGWAY 社の 3D-Coat [79] がボクセルベースのモデリング環境を提供している．しかしこれらの操作は，編集対象としている形状データを書き換えるものであり，編集前の状態は失われてしまう．このように，スカルプトモデラーでの編集操作には可逆性を持たせることは困難であり，アンドゥ・リドゥを実現するには一定の時間間隔でスナップショットを取るしかなく，保持できる操作履歴には限りがある．

一方旧来のモデリングシステムにおいては，ヒストリーベースと呼ばれる手法 [49, 50] が存在する．これは，プリミティブな形状や曲面記述に対して，形状変形を指示するコマンドを蓄積し，組み合わせることで意図した形状を作り出す手法である．編集操作の履歴自体を編集対象として扱うことで，造形工程を自由に遡っ

て編集し直したり，工程の一部分を再利用したりすることが容易であるのが，ヒストリーベースドモデラーの特長である．このヒストリーベースの考え方を画像処理に用いた研究として，Chen ら [97] の研究がある．このようにヒストリーベースで手順を記録することで，コンピュータ上で行う作業の利便性を向上することができる．本研究では，スカルプトモデリングにおいても造形操作を全てヒストリーとして保持することができれば，編集の利便性が大きく向上すると考えた．旧来のヒストリーベースモデラーは，形状変形に逆操作が定義可能なオイラー操作 [51] に基づいており，スカルプトモデラーで用いられるような位相の変化を伴う操作にそのまま適用することはできない．そこで本研究では，スカルプトモデラーにおいて立体造形を行う際，ストロークの入力は視点位置と視線方向によって定義できる平面からの射影によって行う点に注目した．造形操作を行った際の視点情報と，画面平面上でのストロークを保持できれば，造形操作の再現が可能であると考えた．これに基づき，形状モデリング開始時からの全操作履歴を保持した，無制限のアンドゥ・リドゥ操作を実現した．また，形状操作のために入力したストロークを実行時とは異なる形状に適用可能にすることで，直線的に履歴を辿るだけではなく，途中のある工程のみをキャンセルしたり，ある時点から分岐した履歴を適用するなどの，いわゆるツリー構造の履歴情報を扱うことを可能にした．

これらを実現するにあたり，モデリング対象とする形状は Constructive Solid Geometry (CSG) モデル [17, 18] に基づいた陰関数の集合によって表現した．これにより，全ての造形操作は陰関数同士のブーリアン操作 [41, 44] で表現できるため，可逆的なコマンドとして定義することができるようになった．造形のためのストロークを入力するたびに陰関数が増加していくことになるが，陰関数の存在領域をクラスタリングすることによって，インタラクティブな応答速度を実現した．そして，モデリング操作をツリー状に辿ることが可能なインタフェースを実装し，従来のアンドゥ・リドゥに加えて，分岐，任意コマンドのキャンセル，他のコマンドの後ろに別のコマンドを移動・コピーして適用するなどの操作を可能にした．コマンド入力時点と再適用先の形状が変化している場合は「リターゲット」

と呼ぶ処理を導入し，入力平面からの射影を行うことで入力ストロークの意図を活かせるようにした．これらをアプリケーションとして実装し，その動作と実用性を確認した．

4.2 陰関数によるスカルプトモデリングの実現

本節では，陰関数の集合によるスカルプトモデリングを実現するための手法について述べる．編集対象とする形状のデータ構造について述べた上で，スカルプト操作をコマンドとして保持する手法とリターゲット処理について述べる．

4.2.1 形状表現に用いるデータ構造

本章で提案するモデリングシステムにおいては，形状を陰関数表現によるプリミティブの集合演算によって表現する．ここで用いるプリミティブとは，次に述べる要素によって構成するものとする．

- 形状内外判定関数
- プリミティブ属性（ポジティブ・ネガティブ）

各プリミティブが影響を与える領域 P_i を，次の式 (4.1) のように定義する．

$$P_i = \{\mathbf{P} \in \mathbf{R}^3 \mid f_i(\mathbf{P}) \leq 0\} \tag{4.1}$$

ここで，f_i はプリミティブ形状の内外判定を行う任意の関数で，例えば半径 r の球形状プリミティブであれば $f_i(\mathbf{P}) = |\mathbf{P}|^2 - r^2$ となる．任意の 3 次元座標に対して内外が判定できる形状であれば，それらを全てプリミティブとして扱うことが可能である．各プリミティブにはポジティブとネガティブの 2 種類どちらかの属性を設定する．ポジティブは形状の存在する領域を，ネガティブは形状をくりぬく領域を表す．これは CSG モデリングにおける，ブーリアン操作の和と差に相当するものであり，これらをスカルプトモデリングにおける形状の盛りつけと切削操作として用いる．

上記のように定義したプリミティブを用いて，形状全体を表現する集合演算を次のように定める．n 個のプリミティブによって構成した形状モデル S_n は，次のような漸化式 (4.2) によって定義する．

$$\begin{cases} S_1 = P_1 \\ S_i = S_{i-1} \bigoplus P_i \quad (i = 2 \ldots n) \end{cases} \tag{4.2}$$

ここで $\bigoplus$ は次の式 (4.3) に示すように，集合 A, B に対して，

$$A \bigoplus B = \begin{cases} A \cup B & (B \text{ is positive.}) \\ A - B = A \cap \overline{B} & (B \text{ is negative.}) \end{cases} \tag{4.3}$$

を表す条件付き集合演算子である．

モデリングを行うにあたっては，形状表面をインタラクティブに可視化する必要がある．陰関数表現からの等値面抽出処理は，リアルタイムに行うとコストがかかるため，形状表面を表すデータとして，点群の集合によって表現するポイントベースのサーフェス [40, 43, 98] を別途生成して用いることにした．これにより，頂点間の接続関係を考慮せずにユーザの入力操作に対してより直感的な編集結果を提示することができる．ポイントベースのレンダリングを行うにあたり，各プリミティブ形状表面を表す点群 V_i をプリミティブを追加する際に次の式 (4.4) を用いて求める．

$$V_i = \{\mathbf{V} \in \mathbf{R}^3 \mid f_i(\mathbf{V}) = 0\} \tag{4.4}$$

こうして求めた点群 V_i の各頂点に対して，個々の座標が全体形状 S_n の内外どちらに属するかを判定する．追加したプリミティブがネガティブの場合は形状内部だった頂点を，ポジティブの場合は形状外部である頂点を新たな形状表面を表す点群として追加し，該当しない頂点は破棄する．

全体形状 S_n を構成するプリミティブが増加すると，全体形状に対する内外判定関数 $S_n(\mathbf{P})$ の評価にコストがかかってしまうが，このコストの増加は陰関数の存在領域をクラスタリングして領域ごとのリストを作成し，リストの末尾から関数評価を行うことである程度軽減できる．上記の式 (4.4) で述べたように，全体形状 S_n は漸化式によって定義されており，ある座標が形状内部に含まれるか否かはプ

リミティブの関数を順番に評価していき，一番最後にプリミティブ内部に含まれたと判定した際のプリミティブの属性によって決定する．よって，任意の 3 次元座標 $\mathbf{P}$ についてリスト末尾の陰関数から内外判定を行い，最初に内部に含まれたプリミティブがポジティブであった場合は形状内部，ネガティブであった場合は形状外部であると判断できる．該当領域のリストが保持するプリミティブの何れの内部にも含まれなかった場合は，形状外部として扱う．

次の図はプリミティブを追加する際の全体形状の変化と，表面形状を表す点群の追加処理を表したものである．図 4.1 は形状の盛りつけ処理，図 4.2 は形状の切削処理を表す．

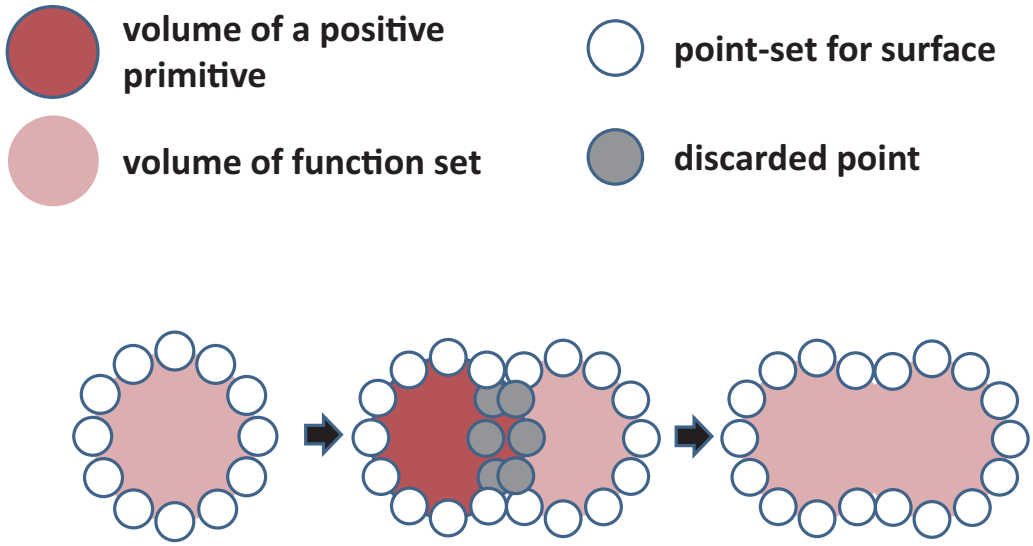


図 4.1: 盛り付けを行っている際のデータ内部状態

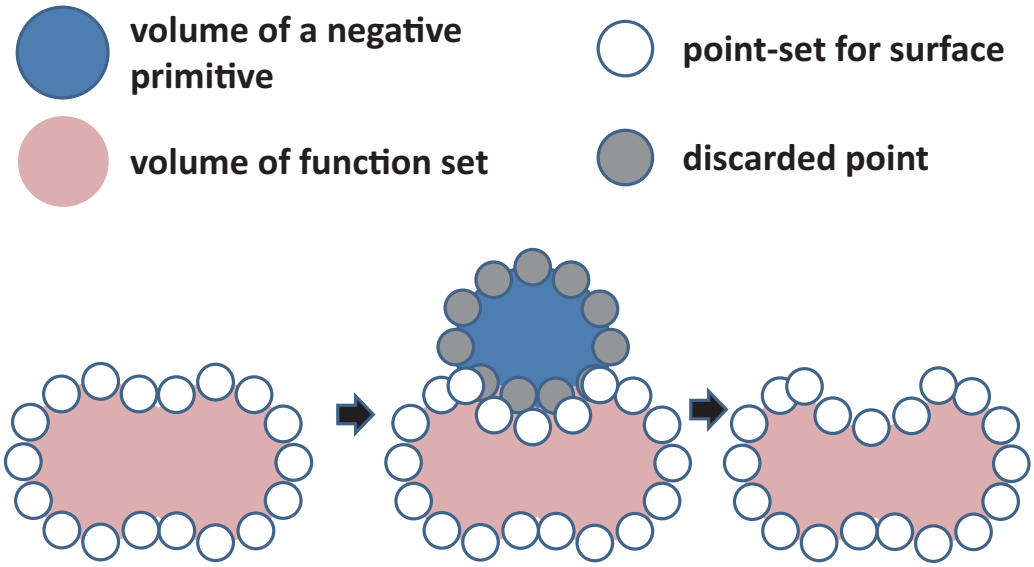


図 4.2: 切削を行っている際のデータ内部状態

4.2.2　ストロークを単位とする履歴管理とリターゲット処理

4.2.1 項で述べたように編集形状は陰関数によって表現するため，スカルプトによる形状の切削や盛りつけは，新たな陰関数を追加することによって実現できる．本章においてはスカルプトモデリングにおける入力を一般的なポインティングデバイスで行うと想定し，1 回のストロークはマウスのボタンやペンタブレットの押し下げを感知してから，押し上げを感知するまでの座標の軌跡によって定義するものとした．これにより，1 ストローク分の履歴情報は全体形状を構成する陰関数リストのうち，該当するストロークの間に追加された陰関数のインデックスを保持することで記録できる．全体形状は陰関数の集合演算として表すため，アンドゥは陰関数リストからの消去，リドゥはリストへの復帰で実現することが可能である．

上記で述べたように，通常のアンドゥ・リドゥは陰関数リストへの追加と削除で実現可能である．本研究ではこれに加えて，履歴の途中に含まれる操作をキャ

ンセルしたり，以前に入力したストロークを任意の状態から再び適用することができれば，スカルプトモデリングにおける編集の利便性が更に向上すると考えた．これを実現するために，入力時と異なる状況下でもストロークを再適用する手法を提案する．本章では，変化した形状に対しても画面から同じストロークを入力したとして処理すれば，ユーザーが入力したストロークに込めた意図を重視した結果が得られると考えた．この処理のことをリターゲットと呼ぶことにする．

リターゲット処理を実現するため，1ストローク分の履歴情報にはストロークをどのカメラ位置から入力したのかも保持しておく．本章における実装では，ストローク入力時の画面座標系における座標列と，視点位置座標，視線ベクトル，視線に対するアップベクトルを記録することとした．この情報を基に，次の流れでリターゲット処理を行う．

- システムの画面をストロークを入力時の状態を再現したカメラワークに設定し，編集形状を描画する．
- 画面座標系上のストローク座標からカメラの視線ベクトルに沿ってレイを飛ばして，編集形状表面と交差した座標を得る．本章の実装においては，グラフィクスハードウェアが備えるデプスバッファの値を参照した．
- ストロークの座標列全てに対して交差点を求め，ストロークのリターゲットを完了する．

図 4.3 は画面上のストロークを入力した際に行う処理の模式図である．図 4.4 における紫色の線は実際に画面上で入力したストロークの軌跡を表し，図 4.5 は形状に適用した様子を表している．

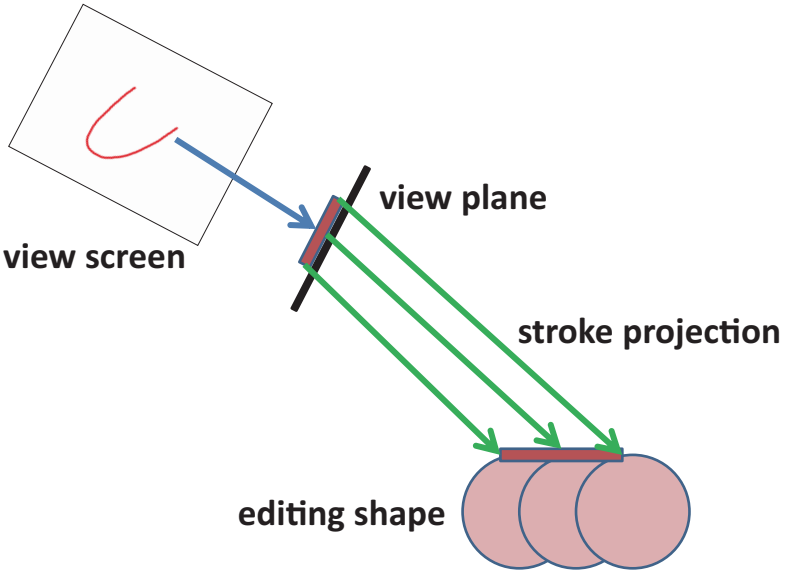


図 4.3: ストローク投影処理の概要

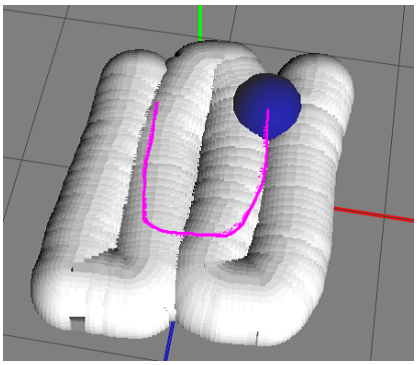


図 4.4: ストローク入力の一例

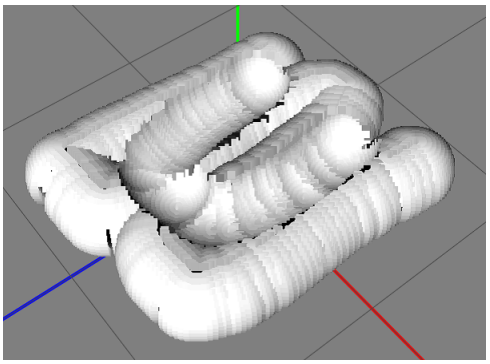


図 4.5: 図 4.4 の入力を形状に適用した様子

上記の図 4.3 で入力したストロークを，履歴操作によって形状が変化した状態で適用する場合の処理を表したのが図 4.6 である．図 4.7 は図 4.4 において入力したストロークを状態が変化している形状に重ね合わせた様子を表し，図 4.8 はリターゲットによって形状に適用した様子を表している．

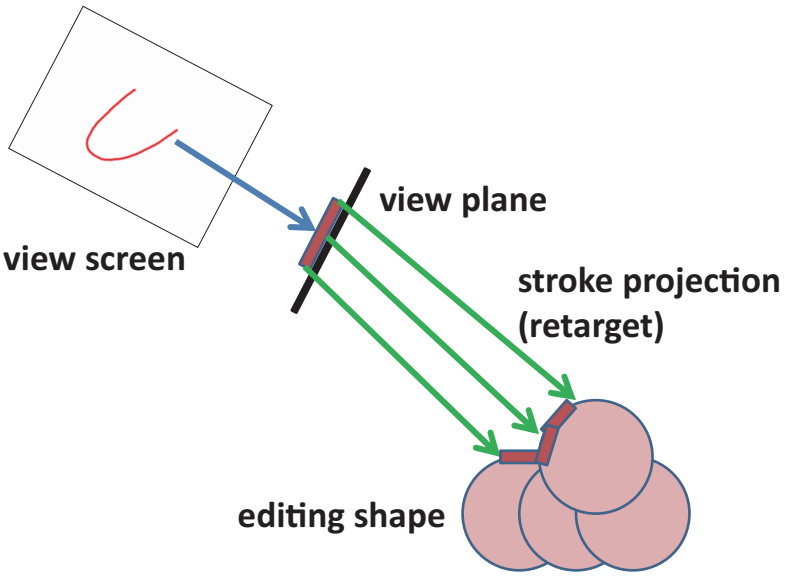


図 4.6: リターゲット処理の概要

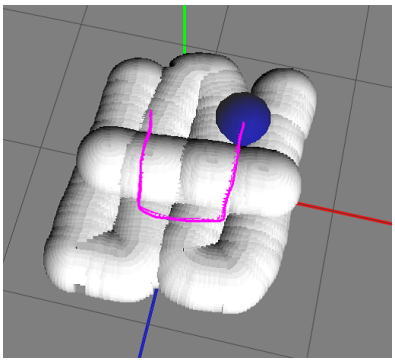


図 4.7: 入力したストロークを変化した形状に重ね合わせた様子

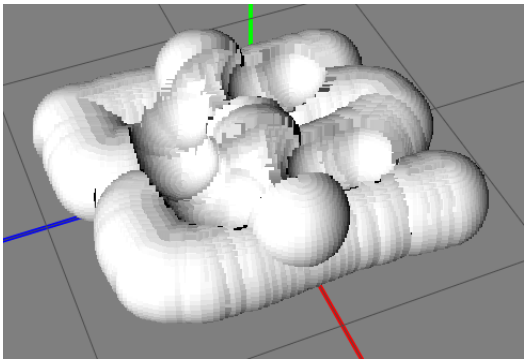


図 4.8: 図 4.7 の入力をリターゲットにより適用した結果

4.3 ヒストリーツリーインタフェースの実装

本章では前節で述べた内容を簡易的なスカルプトモデラーとして実装し，履歴管理に関わる操作を可視化したインタフェースから行えるようにした．これをヒ

ストリーツリーインタフェースと呼ぶ．図 4.9 にインタフェースの概要と，主立った機能の挙動を示す．

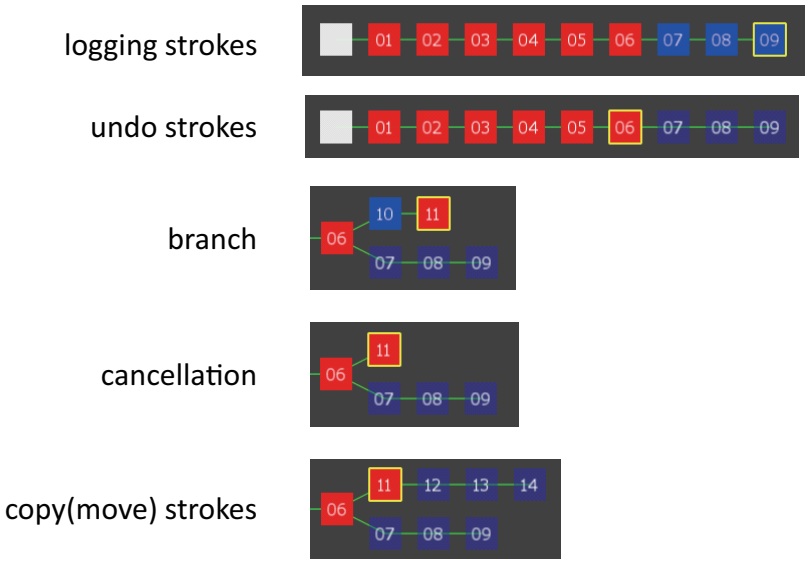
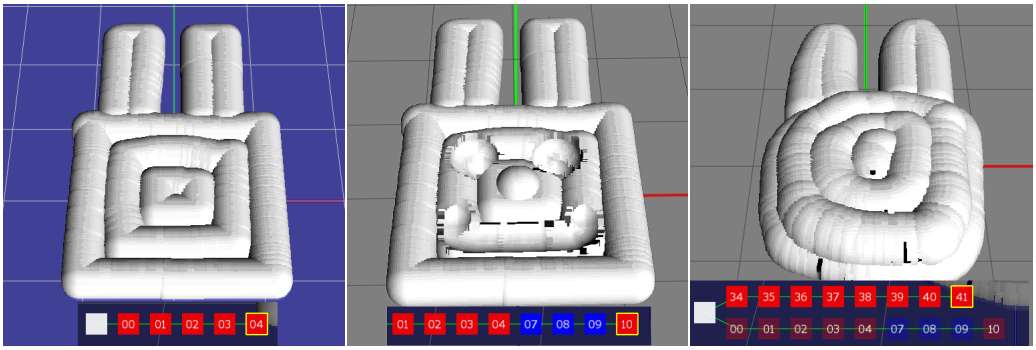
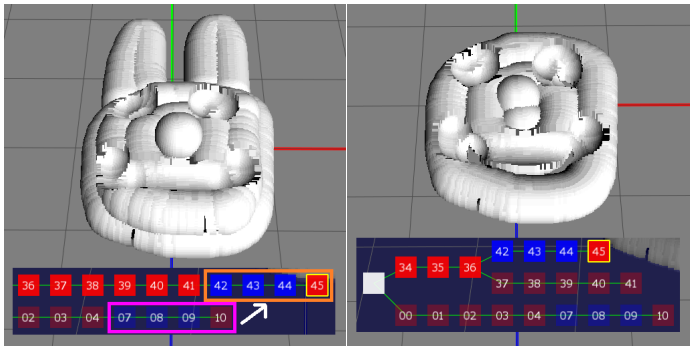


図 4.9: ヒストリーツリーの外観

通常の履歴管理のように，操作を蓄積していくと図 4.9 の上 2 段に示すような表示結果になる．白いアイコンが初期状態を表し，赤と青のアイコンが形状の盛りつけと切削を表す．アンドゥによって履歴を巻き戻した場合は黄枠のカーソルが移動し，キャンセルした操作が半透明で表示される．アンドゥで巻き戻した状態から新たな操作を行った場合，履歴の記録を分岐させて操作情報の蓄積を行う．途中の操作が不要であった場合は，希望する操作だけを取り消すことができる．取り消した操作以降に記録されている操作はリターゲット処理によって再適用する．別の分岐上で行った操作を現状に対して適用したい場合は，アイコンのドラッグ操作によってツリーのコピーまたは移動を行うことができる．コピーや移動を行った操作はリターゲット処理の対象となり，リドゥ時に再適用する．以上のインタフェースによって，本研究によって実現する機能をコントロールできるようにした．



(a) Step 1. (b) Step 2. (c) Step 3.



(d) Step 4. (e) Step 5.

図 4.10: モデリング手順の例

4.4 検証

本システムを C++によって記述したプログラムで実装した．実装に際しては OpenGL ベースの 3DCG ツールキット FK Toolkit system [96] を用いた．

前節で述べた機能を用いて，目的とした操作が実現できるかを検証した．リターゲット処理は操作の途中をキャンセルする場合と，別のツリーにおける操作を適用する場合とで検証を行い，どちらの場合においてもストロークに沿ったスカルプト操作が実現できた．次から示す一連の図はその様子を示す．図 4.10 は，兎のような動物の頭部を模した土台となる形状を作成した様子を示す．

図 (b) は，図 (a) で示した形状に対して目と口の部分に対して切削を行った様子を示している．次の図 (c) は，切削をアンドゥして形状の盛りつけを行ったところを示している．履歴が分岐しており，先ほどの切削操作も記憶していることが分かる．次の図 (d) は，盛りつけを行った後に先ほどアンドゥした切削操作を適用し

たところを示している．リターゲット処理により，状態が変化した形状に対してもストロークが適用されていることが分かる．最後の図 (e) は，リターゲット後に履歴の途中にある操作をキャンセルして，切削を再びリターゲットによって再適用した様子を示している．個々のストロークを保存しておくことにより，適用順序を入れ替えても形状へ反映できることが分かる．

本章における実装では，リターゲット処理を行う際の view plane は入力時の状態のまま固定するものとした．形状が配置も含めて大きく変化した場合は，view plane を随時変更可能なものとすることで，より柔軟なリターゲット処理が実現できると考えられる．

また，陰関数の増加に伴う処理負荷の増加についても検証した．108 回のストロークで 3426 個の陰関数を入力し，約 280000 個の点群によって形状を可視化した状態においても 60FPS で描画と処理がなされることを確認し，速度面でも実用性があると言える．その際の動作環境は次の表 4.1 の通りである．

表 4.1: 動作環境

CPU	AMD Phenom(tm) II X6 1090T 3.2GHz
RAM	4GB
GPU	NVIDIA GeForce GTX 470

今回の実装においては，リターゲットを行う際に編集形状への射影先をデプスバッファの値を用いて求めた．このため，リドゥの動作時に実際に画面描画を行う必要があり，操作に対して結果が反映されるまで若干のタイムラグが発生する．これは射影処理を幾何計算に置き換えることで改善が見込める．

本手法では全ての入力データを保持することにより，スナップショットよりも少ない記憶容量で履歴管理を実現した．この記憶容量についても比較検証を行った．図 4.11 はこの検証のために本システムで作成した形状を示している．最終的な形状を作成するにあたり，入力回数は 8 回，陰関数の個数は 556 個となった．この形状の編集履歴を保持するにあたり，形状全体のスナップショットを取った場合と，

入力データを記録した場合とでの記憶容量の差を表 4.2 と 図 4.12 に示す．

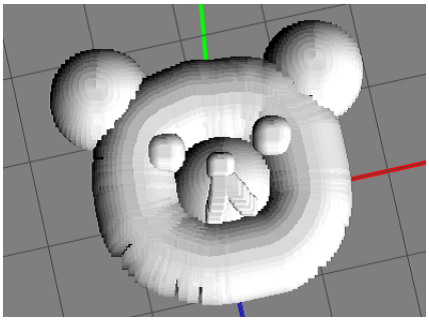


図 4.11: 検証で作成した形状

表 4.2: メモリ使用量の比較

ステップ	ストローク (bytes)	スナップ (bytes)
1	5752	21792
2	6068	43632
3	6384	65520
4	6700	87456
5	7016	109440
6	7332	131472
7	8788	158112
8	9104	184800

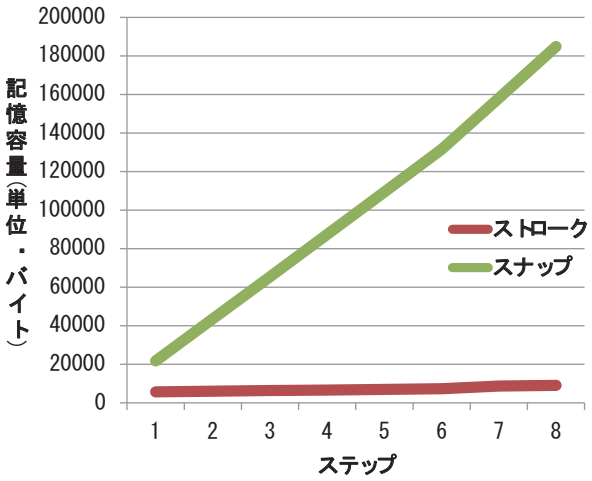


図 4.12: メモリ使用量の比較グラフ

表 4.2 と 図 4.12 に示した通り，入力データを記録することで大きく容量を削減することができた．編集の初期段階においては大きな容量の差は認められなかったが，形状が複雑になるにつれて，入力データを記録する本手法の方が，より省容量で履歴を保持することが可能であることを確認した．実践的なアプリケーションにおいては，スナップショットを取得する手法において，局所的なデータのみを記録するなどの実装が行われるが，その場合は履歴の分岐や編集操作のコピー＆ペーストは実現できなくなる．このため，入力データの再利用性の面においても，ストロークベースの履歴記録には有用性があると言える．

4.5　まとめ

本章では，ストロークの履歴管理に基づいたスカルプトモデリングシステムを提案し，アプリケーションとしての実装を行った．従来のスカルプトモデラーでは制限の多かったアンドゥ・リドゥを無制限に行うことが可能になった．また，履歴の分岐や途中操作のキャンセルも可能とし，編集の利便性が向上した．ストロークの入力時と形状が変化している状態においても，リターゲット処理によってユーザーがスカルプトのために入力したストロークを無駄なく活用することができるシステムを構築できた．

今後の課題として，現状では球形状のみをプリミティブとして扱っているが，任意の凸包や自由曲面を扱えるようにすることで，造形のバリエーションを増やすことが考えられる．また，陰関数の存在領域と表面形状を構成する点群に対してより効果的なクラスタリング処理を導入することで，より複雑で精細な形状の編集にも耐えうるシステムへと発展することが期待できる．

第 5 章

モデリングシステムの実装と運用

5.1 はじめに

本章では，提案手法に基づいて実装したモデリングシステムの機能と得られた操作結果をまとめて，作例を示すことで手法の有用性について検証する．本章で検証の対象とするのは，第3章で述べた点群サーフェスによる形状表現を継承し，第4章で提案したストロークベースのアプローチを盛り込んだスカルプトモデラーの有用性についてである．ストロークベースのモデリング手法を用いて，履歴の操作を利用することでどのようなモデリング工程が可能となるのかについて検証した．

本システムで実現したのは，立体造形を行うストロークベースのスカルプトモデリングである．基本機能としては，スクリーン上で入力ストロークに基づいた形状の盛り付けと切削が可能である．この機能により，何も無い状態から立体物を造形できるほか，既存の形状データを読み込んで編集を加えることも可能となっている．既存の形状データは，表面部分を表す点群データと，形状の内外を表すデータによって構成する必要がある．内外データは，ある座標値が形状の内部か外部かを判定できるデータ形式ならどのようなデータでも良い．一般的には陰関数表現したデータか，ボクセルデータを用いることになる．

この基本機能に加えて，第4章で提案したストローク履歴を保持し，システム側で管理できる機構を搭載している．これにより，操作のアンドゥ・リドゥが無制限に行えるだけでなく，全ての履歴をツリー状のインタフェースで管理し，操作履歴の入れ替えや途中キャンセル・履歴の反復が可能である．

5.2節では，本システムが実現している個々の機能について解説する．5.3節では，本システムの機能において点群を扱う際の内部実装アルゴリズムについて述べる．5.4節では，本システムを用いて形状を最初から生成した場合の動作について検証する．5.5節では，本システムに他のモデリングシステムで作成した形状データを読み込み，編集を加えた場合の動作について検証する．5.6節では検証結果をまとめ，今後の展望について考察する．

5.2　モデリングシステムの機能

本検証で用いたモデリングシステムは次の機能を持つ．

まず最初に必要になるのが，編集形状をその場に生成する機能である．ユーザは，カーソルをマウスやタブレットで操作して，スクリーン上におけるストロークを入力する．このストロークを 3 次元空間中における操作点とすることで，その位置に形状の内部となる領域を陰関数によって生成する．編集操作点の決定方法は，後述する 2 つのアルゴリズムを選択して用いることができる．図 5.1 は，ストロークを入力している様子を表している．図 5.2 は，ストロークにより形状を盛り付けた様子を表している．図 5.3 は，盛り付けた形状を別視点から見た様子を表している．

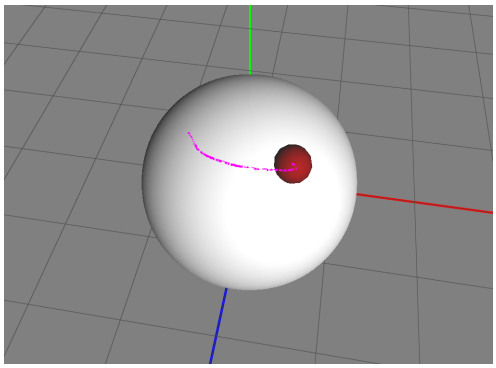


図 5.1: ストロークの入力

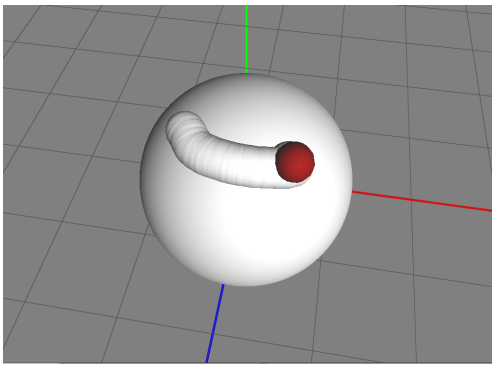


図 5.2: 入力による形状の盛り付け

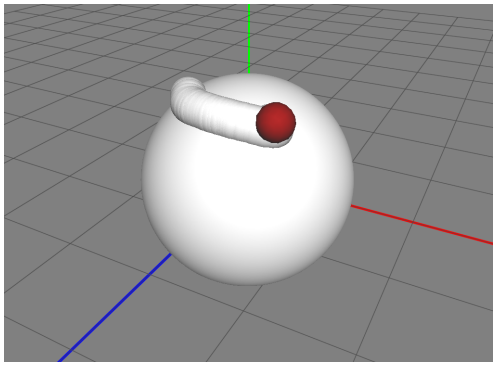


図 5.3: 盛り付けた形状を別視点から見た様子

ツールを切り替えることで，形状に対する切削を行うことができる．盛り付けと同様に，入力したストロークを 3 次元空間中に投影することで，その操作点に形状の差分を取る領域を陰関数によって生成する．図 5.4 は，ストロークにより形状を切削した様子を表している．

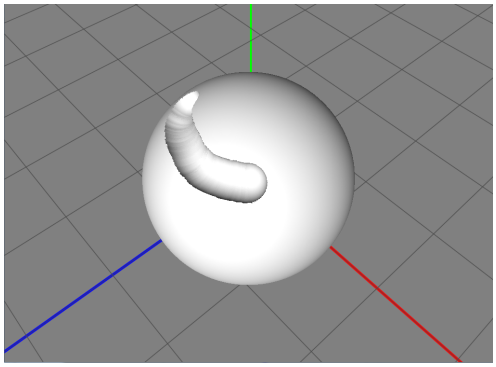


図 5.4: ストロークによる形状切削

形状の盛り付けと切削については，影響半径を任意のサイズに変更することができる．図 5.5 は，半径を変更して切削を行った様子を示している．

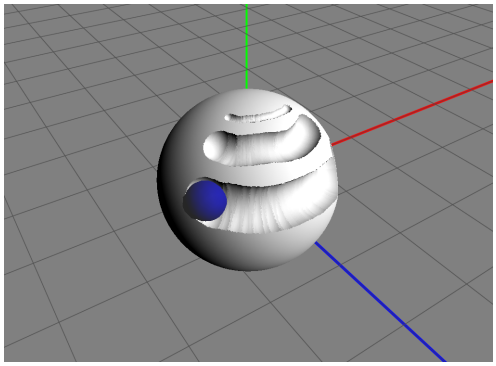


図 5.5: 半径を変更した切削

画面上でのストロークは，3 次元空間中に編集用の仮想平面を定義し，スクリーン平面から仮想平面への射影によって操作点に変換するモードと，編集形状表面に直接投影して適用するモードの 2 種類を選んで運用することができる．図 5.6 は，仮想平面への射影によって盛り付け操作を行っている様子を表している．仮想平

面を用いることで，形状表面の編集に限らない新たな部位の造形や，形状内部のくりぬきが可能になる．図 5.7 は，形状への投影によって盛り付け操作を行っている様子を表している．

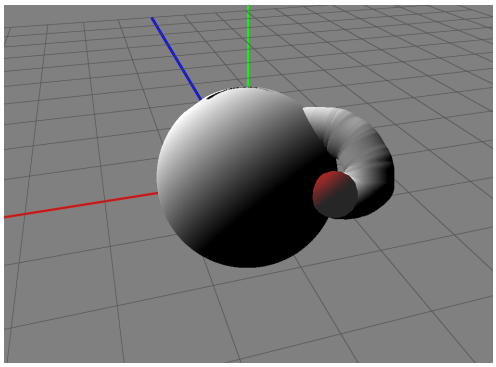


図 5.6: 仮想平面への射影による編集

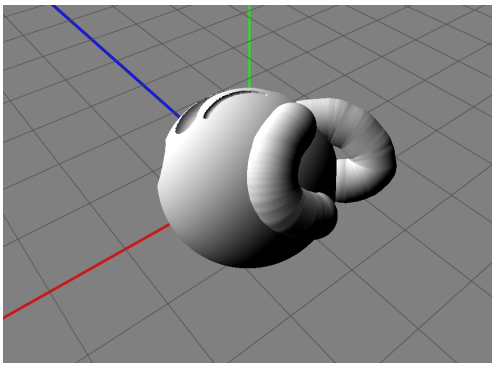


図 5.7: 形状への投影による編集

点群で形状の精細な特徴を表現する場合，頂点の分布密度を高める必要がある．しかし，高密度の点群を常に描画することは，モデリングシステムにおいてはインタラクティブ性を損なうことになる．本システムにおいては，陰関数の表面サンプリング精度を自由なタイミングで変更できるようにした．これにより，ストロークの入力時は低精度で形状の概略を把握し，仕上げの段階で高精度に切り替えて編集するといった利用法が可能になった．精度を低くした場合は個々の頂点の描画サイズを大きくし，精度を高くした場合は逆に描画サイズを小さくすることで，如何なる精度においても形状表面を適切に埋めるようにした．このアルゴリズムについては 5.3 節にて後述する．図 5.8 は低精度で形状を表示している様子を表す．図 5.9 は高精度で形状を表示している様子を表す．

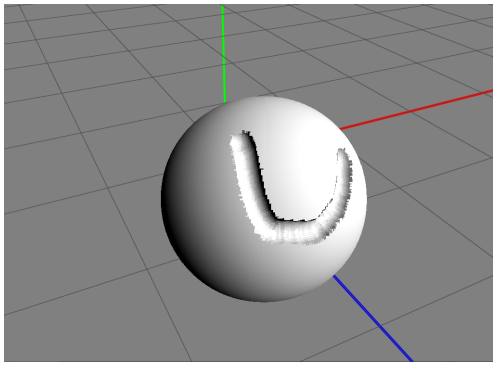


図 5.8: 低精度における形状表示

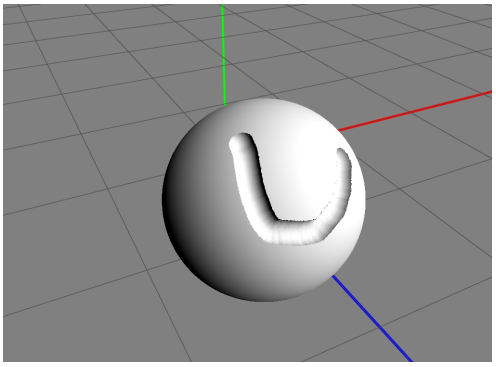


図 5.9: 高精度における形状表示

操作を元に戻すアンドゥと，元に戻した操作をやり直すリドゥは，一般的なアプリケーションと同様に可能であるが，初期状態からの全履歴を保持しているため，無制限に操作を遡ることができる．本システムではユーザの編集操作をアイコンで表現し，それらをツリー状のインタフェースに配置することで，編集履歴を視覚的にも分かりやすく表現した．このインタフェースをヒストリーツリーと呼ぶ．図 5.10 はアンドゥを行った際のヒストリーツリー上における表示内容である．

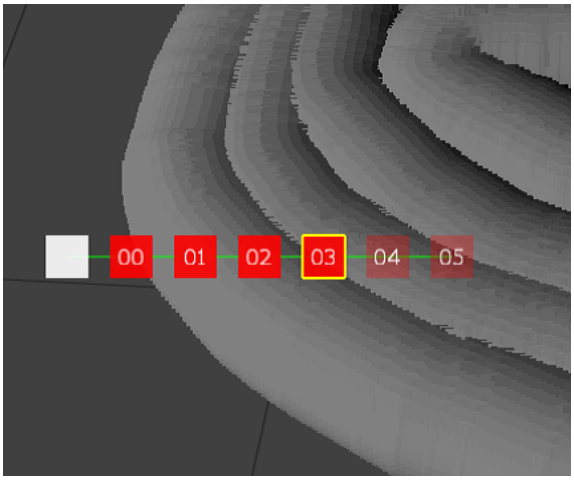


図 5.10: アンドゥを実行中のヒストリーツリー

一般的なアンドゥ・リドゥ操作において，アンドゥ中に新たな編集操作を行った場合は，取り消した操作の内容は破棄される．この制約は，編集中の試行錯誤において煩わしい場合がある．本システムでは，元に戻した操作のうち特定の操作だけをキャンセルし，それ以降の操作については再び適用できるようにした．キャンセルした履歴以降の操作については，ストロークを再び編集形状へ投影，あるいは編集平面に射影する処理を再度行うことで，入力時のニュアンスを極力維持したまま適用することができる．図 5.11 は盛り付けを行った上から切削操作を行った様子を表す．図 5.12 は図 5.11 の状態から，途中で行った盛り付けのみをキャンセルし，切削操作だけ再度実行した様子を表す．

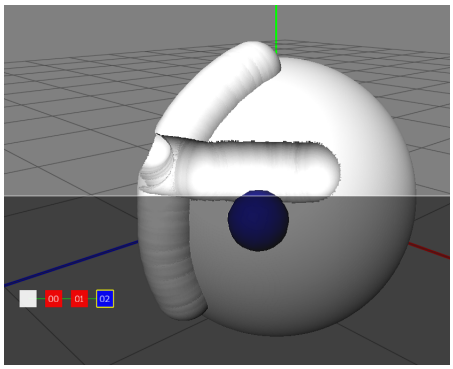


図 5.11: 盛り付けの後に切削を行った状態

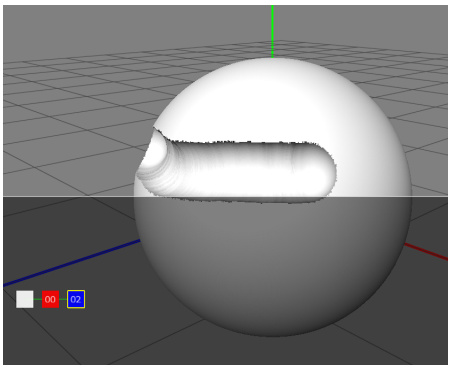


図 5.12: 盛り付けだけをキャンセルした状態

履歴のキャンセル操作を可能にする過程で，入力したストロークを任意のタイミングで再適用することができるようになった．これを応用して，ヒストリーツリーのアイコンをドラッグすることで，一連の操作をコピーできるようになった．例えば，ある土台となる形状に対して行った編集操作を，別の土台形状に対して適用することなどが可能になり，形状モデルのバリエーションを量産したり，他人と協調してモデリング作業を進める際に，ストローク単位でデータをやりとりしたり，後からマージすると言ったモデリングプロセスをスカルプトモデリングにおいて実現することができる．図 5.13 は土台となる元の形状を表す．図 5.14 は，図 5.13 に対して盛り付けのストロークを入力した状態を表す．

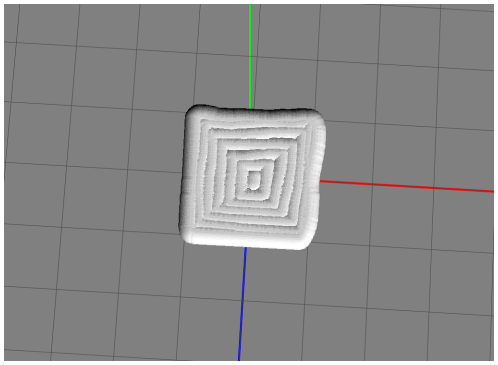


図 5.13: 土台の形状

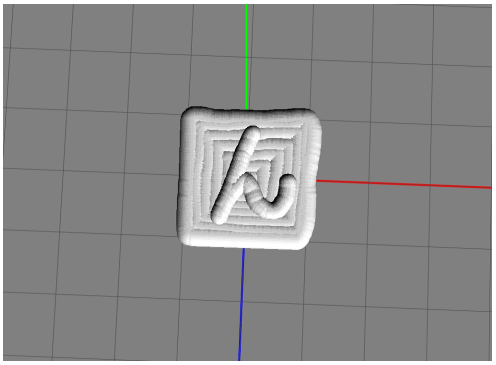


図 5.14: 土台に盛り付けた様子

図 5.15 は，図 5.13 とは異なるノードにおいて別の土台の形状を生成した様子を表す．図 5.16 は，図 5.15 に対して図 5.14 で入力したストロークをコピーし，再適用した状態を表す．

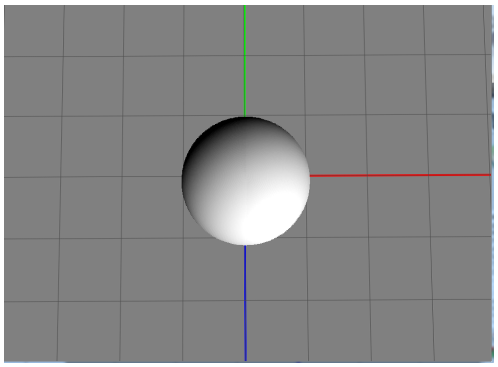


図 5.15: 別の土台の形状

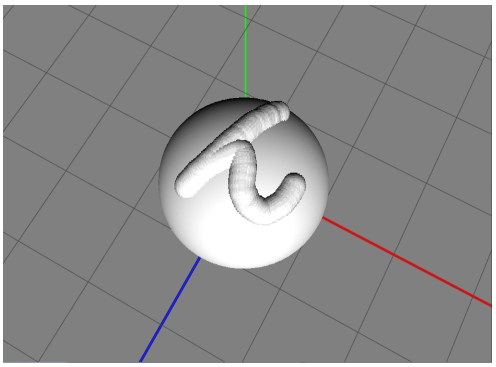


図 5.16: 別の土台にストロークを適用

5.3 点群の生成とレンダリングアルゴリズム

本節では，形状を可視化するにあたって陰関数表面をサンプリングし，点群を生成するためのアルゴリズムについて述べる．本システムでは，形状構成要素として，球と，球をスウィープして生じるカプセル状の領域を取り扱う．まずは球の表面をサンプリングする手法を述べる．

球の中心を示す 3 次元座標を $\mathbf{P}$，球の半径を R，本システムで設定可能なサンプリング精度値を D とする．サンプリング精度値は，3 次元空間における距離単位をどの程度分割するかを表す解像度である．本システムにおいては精度が低い値から順に，$6, 12, 24, 48$ の何れかの値を用いることにした．この時，球面を表す点の座標を得るために三角関数で用いる角度を分割する値 d を次の式 (5.1) によって求める．

$$d = RD \tag{5.1}$$

球面を得るために，球の中心を通る Y 軸方向を中心として円周をサンプリングする．この時，Y 軸方向に応じた円周の拡大率 k_i を次の式 (5.2) によって求める．i は 0 から $2d - 1$ まで 1 ずつ増加する値とする．

$$k_i = R \sin\left(\frac{(i+1)}{2d}\pi\right) \tag{5.2}$$

この k_i を用いて球面を表す座標 $\mathbf{s}_{i,j}$ を次の式 (5.3) によって求める．j は 0 から

$4d$ まで 1 ずつ増加する値とする．

$$\mathbf{s}_{i,j} = \mathbf{P} + \left(k_i \cos\left(\frac{j}{2d}\pi\right), R\cos\left(\frac{(i+1)}{2d}\pi\right), k_i \sin\left(\frac{j}{2d}\pi\right) \right) \tag{5.3}$$

以上の手続きによって球面座標のサンプリングを行う．カプセル状領域の場合は，前述した手続きにおける球の中心座標を，球の領域を構成する 2 点間を距離 $\frac{1}{D}$ ずつ移動しながら実行することでスウィープした領域の表面を得る．その際，既に処理済みの領域内に含まれている点群はカプセル内部の点にあたるので破棄する．

このようにして生成した点群を，4 章の 4.2.1 項で述べた式に従って可視化する際に用いる座標とするか否かを判定する．

本システムではサンプリング精度値を変更することで，インタラクティブな応答速度と形状を可視化する際の精細度を調整することが可能であるが，レンダリングの際には点と点の間を適切に埋めるように点の描画サイズを調整する必要がある．本システムは，点群の描画に際してシェーダープログラムを用いており，その中で次の式 (5.4) による点の描画サイズ調節を行っている．w は，点をレンダリングする時にスクリーン上を占めるピクセル幅を表す．C は任意の定数であるが，本システムにおいては 48 が最も適切な数値であった．z はシェーダー内部で算出する，カメラから点までの正規化した距離を表す．$\min(A, B)$ は A か B のうち，小さい方の値を返す関数である．

$$w = \min\left(\frac{C^2}{D}, \frac{C^2}{zD}\right) \tag{5.4}$$

このようにして，カメラとの距離に応じて点の描画サイズを調整し，適切に間隔を埋めて描画することが可能になった．

5.4 モデリングシステムによる作例

本節では，実際にキャラクターをモデリングし，その手順を操作することでバリエーションを展開した様子を解説し，ストロークベースの手法の有用性を検証する．ここではプリミティブな形状を組み合わせてキャラクターを模した形状を

モデリングする目的を想定している．その上で，造形途中の試行錯誤やバリエーションパターンの検討を行いたいと思う場面は多い．本節では，履歴の操作によってこの試行錯誤を支援できるかどうかを検証する．

図 5.17 は，キャラクターを模した形状をモデリングした様子を示している．

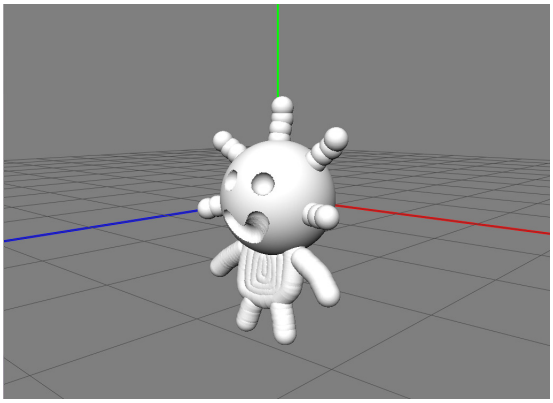


図 5.17: キャラクターのモデリング

表情の造形を検証するために，アンドゥで造形操作を取り消して異なる表情をストロークで入力することができる．図 5.18 は，表情を造形したストロークをいったんキャンセルした様子を示している．ヒストリーツリーのアイコンが巻き戻っているのが分かる．

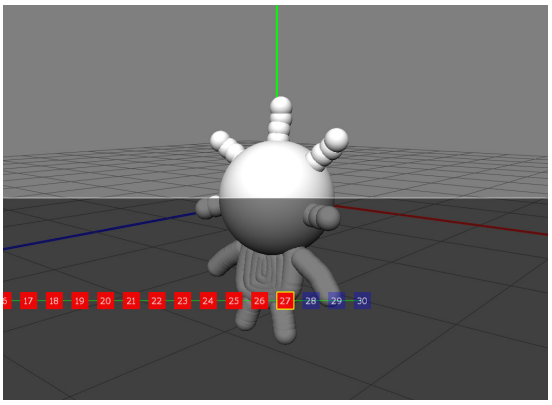


図 5.18: 表情の造形をキャンセル

キャンセルしたところへ再度ストロークを入力する．図 5.19 は，表情を造形し

直した様子を表している．

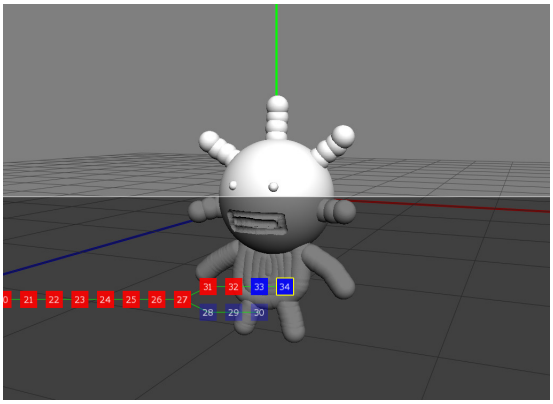


図 5.19: 顔の表情を差し替え

造形を行っていった結果，大元の土台の形状が気に入らなかったとする．このような場合は，初期状態までアンドゥした状態で新たな土台の形状を作成することができる．図 5.20 は，新たに作成し直した土台の形状を示している．

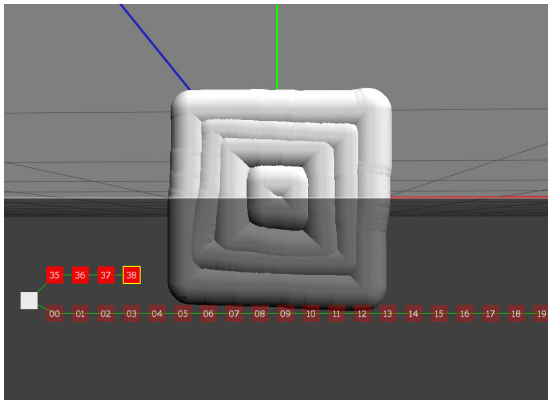


図 5.20: 土台を作り直した様子

新しく土台の形状を生成した後に，これまでに造形した操作をコピーして適用することで，これまでの入力を活かしつつ土台の形状のみを入れ替えた状態の形状を生成することができた．図 5.21 は，新しく生成した形状モデルを表している．

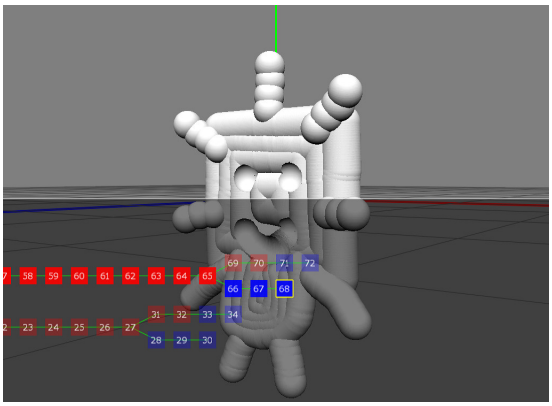


図 5.21: 土台のみ入れ替えてストロークを適用

新たに生成した形状に対して，表情の別バージョンを適用することもできる．図 5.22 は，表情の別バージョンを適用した様子を表している．

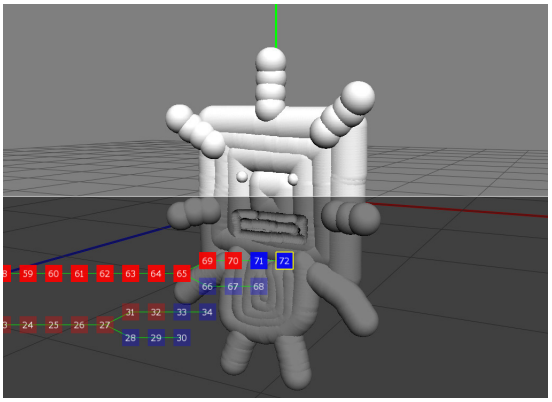


図 5.22: 土台を入れ替えて別表情を適用

このように，造形物のバージョン違いを容易に作成し，切り替えることが可能になった．個別にファイル保存するような運用面でのカバーとは異なり，全ての操作がシステム上でシームレスに行うことができ，有用性を示せたと考えられる．

5.5　モデリングシステムで既存のモデルを編集した例

本節では既存のモデルを本システムに読み込み，形状の編集操作を行った例を示す．本研究で提案する手法は，データの変換を経ることで既存の形状データと

併用することが可能である．この検証においてはティーポットのモデルデータから内外情報を表すボクセルデータと，形状表面を表す点群データを生成して用いた．本手法が既存の形状データの編集においても有効であるかを検証し，本研究の有用性を示す．図 5.23 は，システム上に読み込んだティーポットを表示した様子を示している．

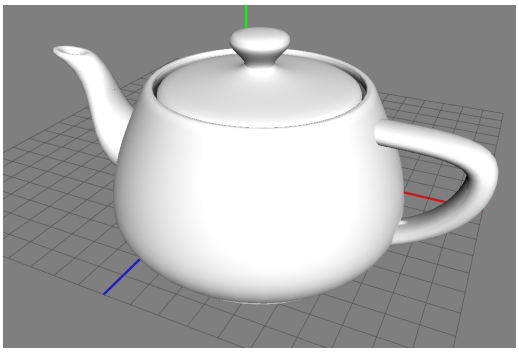


図 5.23: ティーポットのモデルを表示した様子

このモデルに対し，文字の彫りつけ，形状の盛り上げ，取っての形状追加などの操作を行った．図 5.24 は，ティーポットに文字を彫刻した様子を示している．図 5.25 は，ティーポットの表面を盛り上げた様子を示している．図 5.26 は，図 5.24 における文字の彫刻を図 5.25 に示した盛り上げ操作の後に適用した様子を示している．図 5.27 は，ティーポットの取っ手に新たな部位を造形して追加した様子を表している．いずれの操作も問題なく実行でき，本研究で提案した手法が既存のモデルデータとのコラボレーションにおいても有用であることが証明できた．

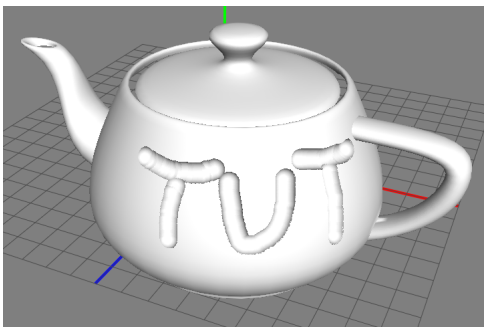


図 5.24: ティーポットへの文字彫刻

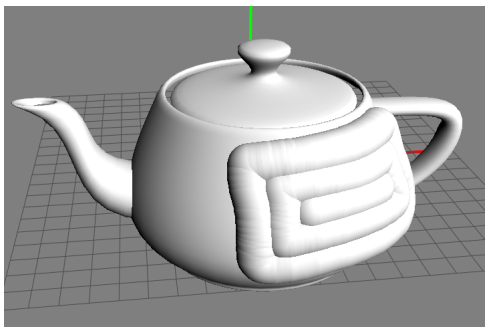


図 5.25: ティーポット表面の盛り上げ

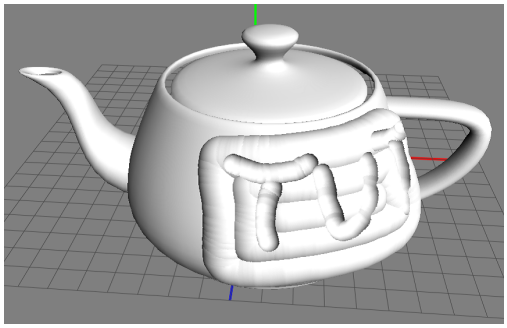


図 5.26: 盛り上げた上に文字彫刻を適用

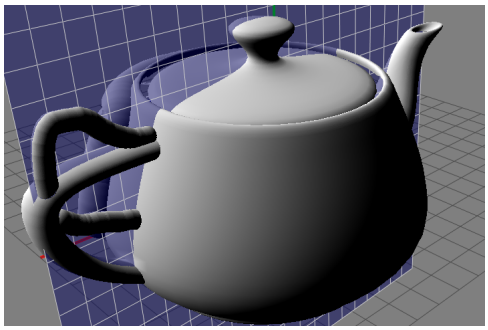


図 5.27: 取っ手に形状を追加

更に，一連の造形操作を行った様子を示す．図 5.28 は最初に行った盛り上げ操作の結果を表している．赤いアイコンの 00 から 02 までが該当する．

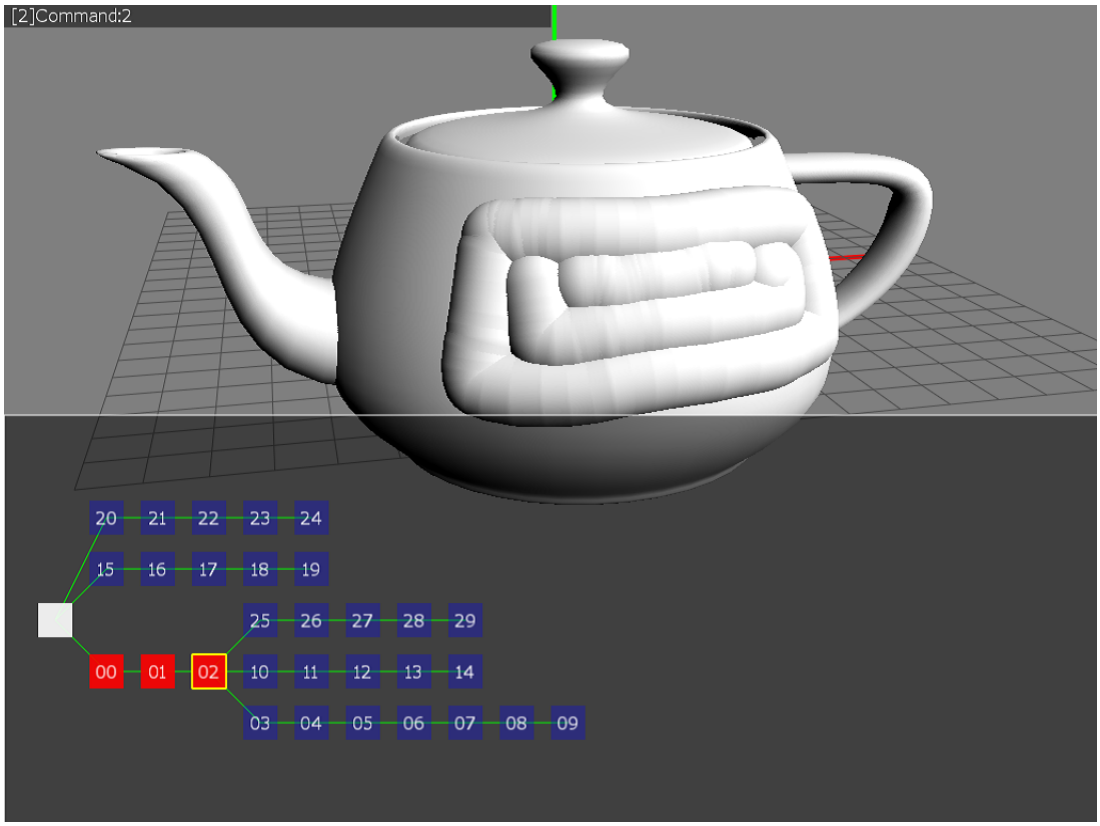


図 5.28: ティーポット表面の盛り上げ操作のログ

次に，盛り上げた部位に感情を表す顔文字を彫刻した様子を図 5.29 に示す．青いアイコンの 03 から 07 までが該当する．08 と 09 は望ましくない結果であったため，キャンセルした．

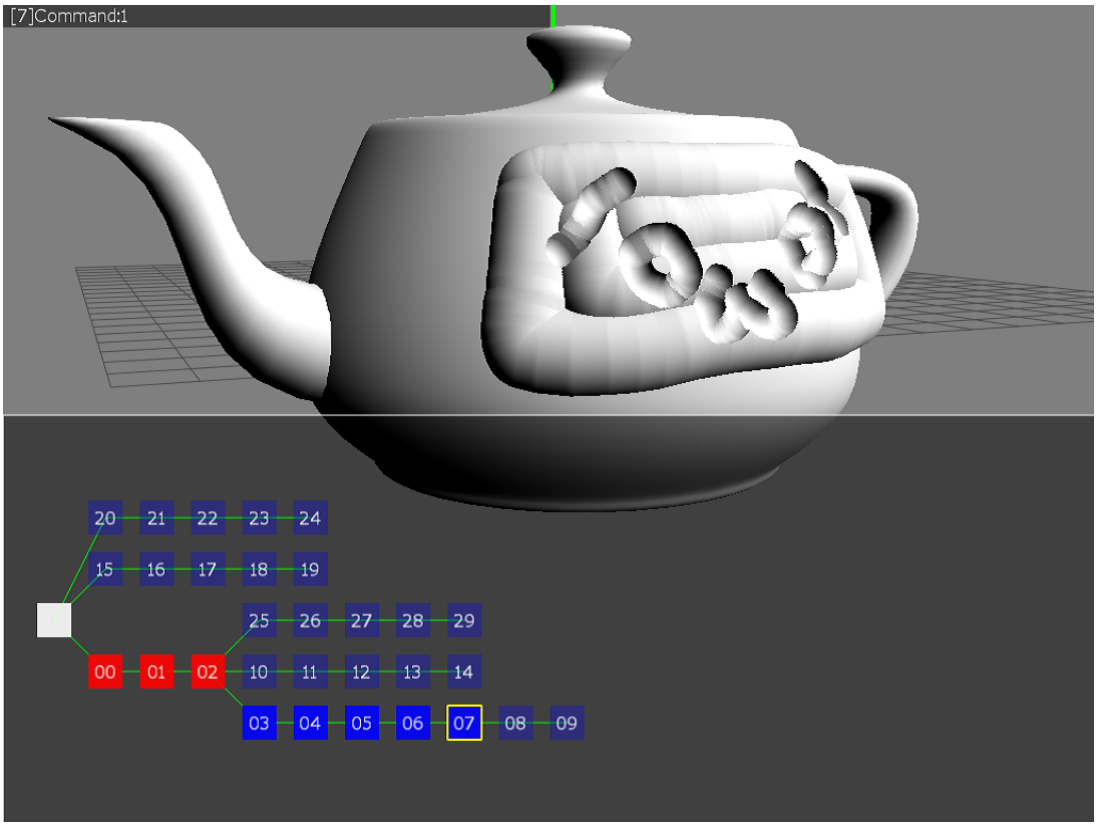


図 5.29: 盛り上げ部位に顔文字を彫刻した操作のログ

次に，目となる部位の彫刻をやり直して適用した様子を図 5.30 に示す．青いアイコンの 10 から 14 が該当する．更にこの結果を，盛り上げ操作を行わずに直接ティーポット表面に適用した結果を図 5.31 に示す．これは 10 から 14 までの操作を編集開始直後にコピーして実現しており，青いアイコンの 15 から 19 が該当する．

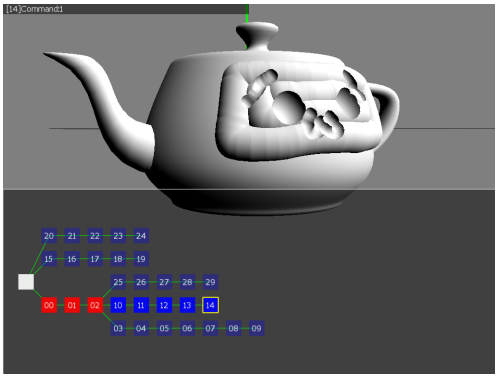


図 5.30: 目の造形をやり直した様子

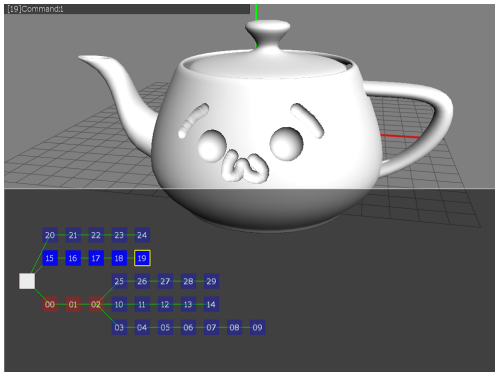


図 5.31: 顔文字の彫刻を表面に適用

次に，顔文字の表情を変化させた様子を図 5.32 に示す．青いアイコンの 20 から 24 が該当する．この操作も 15 から 19 までの操作を複製し，一部操作をキャンセルして差し替えることによって実現している．更にこの造形結果を形状の盛り上げ操作後に適用した様子を図 5.33 に示す．これは 20 から 24 の操作を 02 の操作後にコピーして実現しており，青いアイコンの 25 から 29 が該当する．

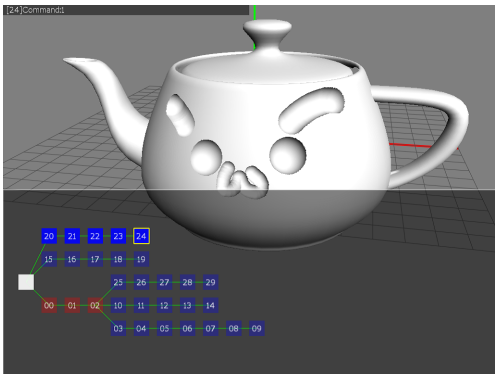


図 5.32: 顔文字の表情を変化

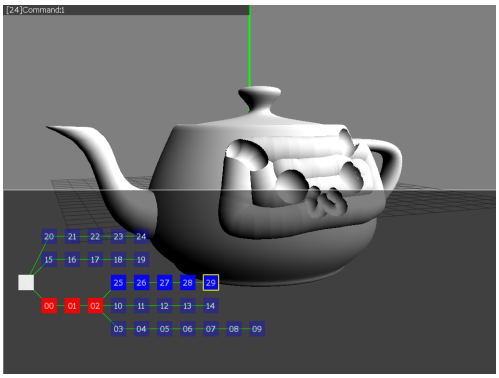


図 5.33: 盛り上げた形状の上に変化を適用

このように，ある造形操作を何度も利用してモデリングを行うことが可能になった．形状が変化した後に造形操作を適用するだけではなく，操作履歴をツリー状に管理していることで造形のバリエーションを同時に複数保つことができるため，形状の試作パターンをいくつも試したり，異なるパターン同士のコラボレーションが容易に行えるようになった．

更に，本実装におけるメモリ使用量の効率化について，複数の形状においての比較を行った．表 5.1 と 図 5.34 は，図 5.24 の作例におけるメモリの使用量を，ストロークベースとスナップショットベースで比較した結果である．同様に，表 5.2 と 図 5.35 は，図 5.26 の作例における比較結果を，表 5.3 と 図 5.36 は，図 5.30 の作例における比較結果を示す．

表 5.1: メモリ使用量の比較 (1)

ステップ	ストローク (bytes)	スナップ (bytes)
1	580	808
2	1100	2264
3	1920	5168
4	2380	8560
5	2900	12600

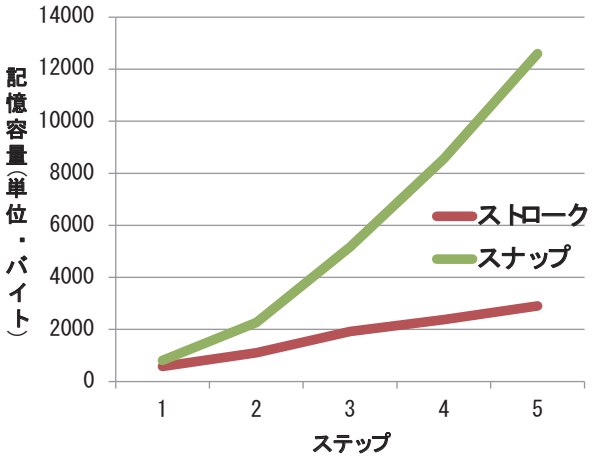


図 5.34: メモリ使用量の比較グラフ (1)

表 5.2: メモリ使用量の比較 (2)

ステップ	ストローク (bytes)	スナップ (bytes)
1	2548	6056
2	5024	17976
3	5868	31408
4	6448	45648
5	6968	60536
6	7788	76872
7	8248	93696
8	8768	11168

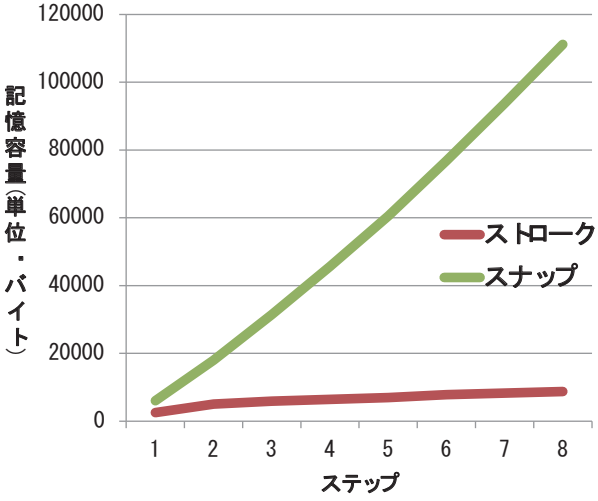


図 5.35: メモリ使用量の比較グラフ (2)

表 5.3: メモリ使用量の比較 (3)

ステップ	ストローク (bytes)	スナップ (bytes)
1	1180	2408
2	2096	6520
3	2832	11856
4	3316	17744
5	3968	24632
6	4392	31912
7	5044	40192
8	5780	49696
9	6096	59304
10	6412	69016

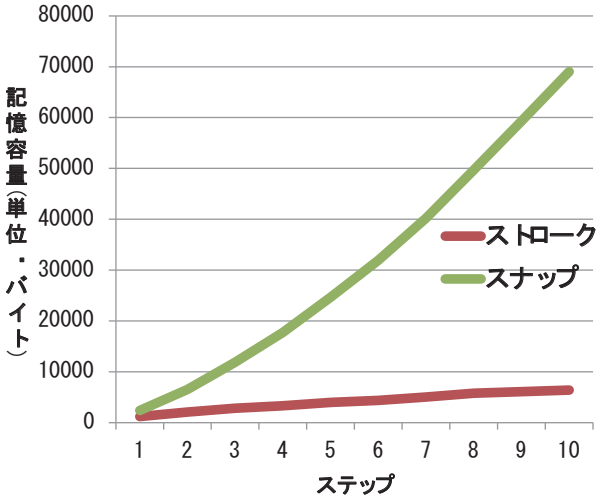


図 5.36: メモリ使用量の比較グラフ (3)

何れの作例においても，入力データを記録することで大きく容量を削減することができた．本システムでは形状の状態をツリー状に管理することができるため，1 つ 1 つの状態を少ない記憶容量で保持できることは重要である．ストロークベースのデータ管理を用いることで，3 次元形状のバリエーションを多数保持し，バージョン管理的なアプローチによるモデリングが可能になったと言える．

5.6 まとめ

これまでの機能と作例により，ストロークベースのモデリング手法の実際と，その有用性を検証した．入力したストロークの履歴を入れ替えたり，複製することで形状をモデリングしていくことが可能なことが分かった．ストロークをベースにすることで，一度入力したユーザの筆致や軌跡を何度も利用することが可能になった．また，履歴をツリー状に可視化し，ストロークの適用順序を自由に入れ替えたり複製したりすることで，様々な形状のバリエーションを同時に保持したまま編集していくことが可能になった．これはユーザの試行錯誤を支援したり，バリエーションを保持したまま造形作業を行うことが可能な新しいモデリングシス

テムを実現できたと言える．

今後の展望としては，複数ユーザーが編集したストロークの履歴を統合して管理することで，複数人で1つの造形物を作成する共同作業にも応用することが考えられる．これまでにも，異なるユーザーが作成した造形部品を組み合わせるモデリングが可能なシステムは多い．しかし，ストロークの履歴レベルでの組み合わせが可能になることで，ユーザーが形状に対して施した造形のニュアンスや，細かいタッチを再利用して統合するようなワークフローが確立できる．図5.37に，操作履歴とそれに基づいた編集結果をツリー状に記憶した，本システムのイメージ図を示す．図中のアルファベットはストロークヒストリーによって生成した各バージョンを表す．”retarget”と表記した矢印は，各バージョンにおける状態を，任意のバージョンが示す状態に対して適用することで，履歴操作の再適用によって形状を生成したことを示している．図中では，リビジョンaに対してbの操作を複合したリビジョンa+bを生成したり，リビジョンd,eの状態を初期状態に対して適用したバージョンを生成すると言った操作を実現している様子を表している．将来的にはこのようなインタフェースを実装することで，より直感的な履歴操作が可能になると言える．

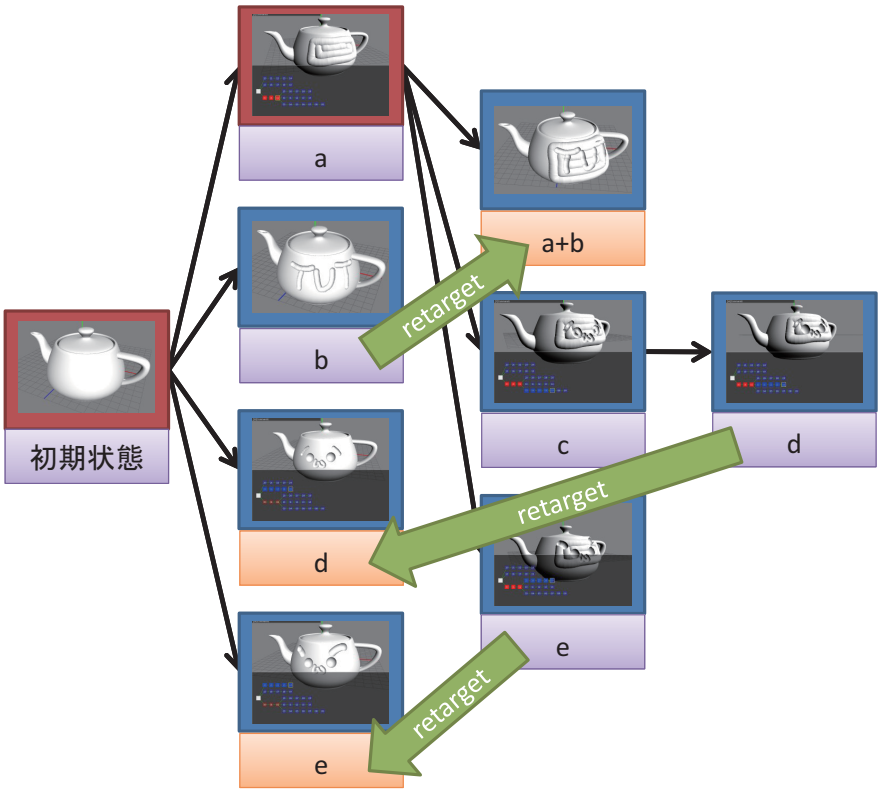


図 5.37: モデリングプロセスをヒストリーツリーで表した図

第 6 章

結論

6.1 研究のまとめと成果

本研究は，既存の 3 次元形状モデリングに対する問題点を指摘し，2 つのアプローチによってこれを解決したものである．

第 1 章と第 2 章で指摘した問題点は次の 2 点である．まず 1 つは，GPU の発展過程によってリアルタイムグラフィクスにおいて用いるデータ構造がポリゴンとテクスチャを主としたものになっており，既存のモデリングツールがポリゴンを主としたデータ構造に偏っていること．そして 2 つめには，複雑な形状編集操作が生成するデータのみに注目しているため，編集履歴が非可逆的な構造になっており，自由なアンドゥ・リドゥや編集履歴の再利用が困難であることである．

これらの問題点に対する提案として，第 3 章では点群を直接編集することによるモデリングシステムを構築した．点群を用いたスカルプト操作が持つメリットは，ポリゴンのように位相の接続関係を気にせずとも編集ができる点と，点群の密度を調節することで LOD が実現可能なことが挙げられる．さらに第 3 章での提案では，画像のシルエットと色情報を編集形状と点群の色に反映することで，画像の特徴に基づいた初期形状を簡易に制作することも可能になった．

また第 4 章では，モデラーのユーザが入力したストロークに注目し，このデータを保持することで完全なアンドゥ・リドゥが可能なスカルプトモデラーを構築した．2 次元平面でのストローク入力情報を保持することで，3 次元形状のスナップショットを保持し続けるよりも遙かに低容量で履歴を保持することが可能になった．編集する 3 次元形状は，ストローク情報を 3 次元空間内に投影することで陰関数集合を構築し，CSG モデルを用いたブール演算で表現した．これにより，履歴を一本道に辿るだけではなく，別の工程で入力したストロークや，複数人で行った編集作業をマージするといった，バージョン管理に近い 3 次元形状モデラーを実現した．

第 5 章では，ここまでに述べた成果を用いてシステムを運用した結果の検証を行い，第 3 章および第 4 章で述べた手法の有用性を確認した．

本研究によって得られた成果をまとめた結果，以下の結論が得られた．

1. 解像度や位相構造を考慮せずに自由形状をモデリングできるようになった．
2. 画像データから点群の集合による編集形状を作成し，容易に修正ができるようになった．
3. スカルプトモデリングにおいて，無制限にアンドゥ・リドゥ操作ができるようになった．
4. 履歴を操作することで，複数人で手分けして編集した形状をマージしたり，編集手順の入れ替えや反復によって形状生成することができるようになった．

以上の結果，本研究で目的としていた，3次元形状モデリングにおけるデータ構造とその操作に関わる問題を解決し，新たなモデリングのアプローチを確立することができた．

6.2 今後の課題と展開

本研究により，3次元形状モデリングに対してストロークベースという新たなアプローチを確立することができた．しかし，このアプローチを最大限に活かすためには，現状から大きく改善しなくてはならない点や，新たに確立するべき手法が存在する．

まず，陰関数集合から形状表面を抽出する処理を高速化することが必要である．現状の実装では一切の最適化を行っていないため，点群の密集度を上げた状態ではストローク入力から形状生成までに処理待ち時間が生じてしまう．この点については，GPGPU による並列化を導入することで，大幅な処理時間の削減が見込める．

また，現状ではアンドゥの処理を形状編集開始時からの全履歴を再トレースすることで実現しているため，ステップ数がかさんだ状態でのアンドゥにはやはり待ち時間が生じてしまう．これは，直近の形状に関してはスナップショットを取る

などして，従来手法とのハイブリッドな実装によって回避することが可能であると考えられる．

本研究では，従来のスカルプトモデリングで実現できている操作のうち，形状の盛り付けと切削というプリミティブな編集のみを対象としている．一般的なモデラーでは，押し潰しや伸張，平滑化などの編集操作も良く用いるため，これらの操作もストロークベースで対応できるように，手法を拡張していくことが望まれると考えられる．

更に，本研究で確立した「ストロークそのものを保持する」というアプローチの利点を活かすために，形状そのものではなく入力したストロークを後から編集することで，形状を間接的に修正，編集する手法が考えられる．Teddy のようにスケッチ入力をトリガーとするだけではなく，その後も保持し続けて編集できるようにすることで，2 次元の入力が 3 次元形状に及ぼす影響を明確になり，よりユーザの入力を活かしたモデリングが可能になると考えられる．

謝辞

本研究は長年にわたり，多くの方々のご指導・ご支援があって完遂することができました．学部・修士から引き続きまして研究の指針から開発の手法，論文の執筆と幅広いご指導ご教授を頂きました，渡辺大地先生に感謝いたします．公私にわたり，様々な場面でお力添えを頂きました．また研究に対し違った切り口からの示唆を頂くと共に，発表の場や教壇に立つ機会を与えてくださいました，三上浩司先生に感謝いたします．渡辺先生と共に，教員として，時には兄のように接して頂きました．厚く御礼申し上げます．

主査をお引き受け下さいました，近藤邦雄先生に感謝いたします．プレッシャーに負けそうな私を常に気に掛けて見守って頂きつつ，ご指導頂きました．そして，研究主査をお引き受け頂きながら，数年にわたってお待たせしてしまいました，副査の宮岡伸一郎先生に感謝いたします．点群というアプローチに共感を頂いたことは大きな後押しになりました．本物を志向するというマインドは，今後の人生においても忘れません．審査にあたり，ご多忙の中副査をお引き受けくださいました，飯田仁メディアサイエンス専攻長，柿本正憲先生，千代倉弘明先生に感謝いたします．メディア学部2期生として入学して以来，多くの先生方にお世話になりました．中でも，故渕上季代絵先生からご教授頂きました「数式が作るかたち」という視点は，この研究テーマを定めるにあたり欠かすことのできないものでした．ありがとうございました．

そして，博士課程という戦いの場において，戦友として共に戦い，励まし合った同課程の渡邉賢悟さんに感謝したいと思います．共に戦う仲間が居ることが，どれだけ心の支えになったか分かりません．研究を進めるにあたって様々な意見を交換してくれた，本学メディア学部のゲームサイエンスプロジェクトの歴代メンバーに感謝します．良い先輩で居続けられた自信はありませんが，皆さんがこの研究室に来てくれて本当に良かったと思います．論文執筆を進めるにあたり，実に示唆に富んだディスカッションと素敵なカレーを共にしてくれた，メディア学研究会メンバーに感謝します．これからもカレー食べましょう．研究や授業で疲れた時に，美味しいおそば＆天ぷらと素敵な笑顔で癒していただきました，伊奈

喜のおっちゃんに感謝します．いつもごちそうさまでした．日々の暮らしに疲れた時には，本学厚生施設のフーズフー裏にて私に癒しをくれました，本学の守り猫・どん様に感謝を捧げたいと思います．今後とも，学舎をお守りくださいませ．学内外にわたり，本研究にご協力いただきました全ての皆様と，この論文に目を通してくださった全ての方々に，厚くお礼を申し上げます．

いつも私を支えてくれた両親と，全ての友人たちに感謝します．誰一人とて欠けては辿り着けない道でした．また，孤独な作業の中で私に力と勇気を与えてくれた,「世界樹の迷宮 IV」のサウンドトラック・アレンジアルバムを制作してくださったスタッフの皆さんに感謝します．私の身体にはゲームミュージックが血潮として流れているのだということを再確認できました．

そして最後に，在学中からずっと変わらず私を支えてくれている，妻の佳衣に感謝します．ありがとう．

参考文献

[1] James D. Foley(著), Steven K. Feiner(著), AndriesDam (著), John F. Hughes(著), 佐藤 義雄 (翻訳). コンピュータグラフィックス理論と実践. オーム社, 2001.

[2] ロイ・A. プラストック, ゴードン・カレイ. マグロウヒル大学演習シリーズ コンピュータグラフィクス. マグロウヒル出版, 1987.

[3] 中前栄八郎, 西田友是. 3 次元コンピュータグラフィクス. 昭晃堂, 1997.

[4] M. ドバーグ, M. ファン. クリベルド, M. オーバマーズ, O. シュワルツコップ. コンピュータ・ジオメトリ　計算機科学：アルゴリズムと応用. 近代科学社, 2000.

[5] Langetepe Elmar and Zachmann Gabriel. 空間的データ構造とアルゴリズム. 株式会社ボーンデジタル, 2007.

[6] Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pp. 115–122, 1998.

[7] Tomas Akenine-Moller and Eric Haines. リアルタイムレンダリング 第 2 版. 株式会社ボーンデジタル, 2006.

[8] Ron Fosner. *Real-Time Shader Programming*. 株式会社ボーンデジタル, 2003.

[9] Matthias Muller, Matthias Teschner, and Markus Gross. Physically-Based Simulation of Objects Represented by Surface Meshes. In *Proceedings of the Computer Graphics International*, CGI '04, pp. 26–33, Washington, DC, USA, 2004. IEEE Computer Society.

[10] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. FiberMesh: designing freeform surfaces with 3D curves. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[11] Caiyun Yang, H. Suzuki, Y. Ohtake, and T. Michikawa. Boundary smoothing for mesh segmentation. In *Computer-Aided Design and Computer Graphics, 2009. CAD/Graphics '09. 11th IEEE International Conference on*, pp. 241 –248, aug. 2009.

[12] Ryan Schmidt and Karan Singh. meshmixer: an interface for rapid mesh composition. In *ACM SIGGRAPH 2010 Talks*, SIGGRAPH '10, pp. 6:1–6:1, 2010.

[13] Matthieu Nesme, Paul G. Kry, Lenka Jeřábková, and François Faure. Preserving topology and elasticity for embedded deformable models. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pp. 52:1–52:9, New York, NY, USA, 2009. ACM.

[14] GERALD E.FARIN. NURBS 射影幾何学から実務まで 第 2 版. 共立出版, 2001.

[15] Thomas J. Cashman, Ursula H. Augsdörfer, Neil A. Dodgson, and Malcolm A. Sabin. NURBS with extraordinary points: high-degree, non-uniform, rational subdivision schemes. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pp. 46:1–46:9, New York, NY, USA, 2009. ACM.

[16] Andre Schollmeyer and Bernd Fröhlich. Direct trimming of NURBS surfaces on the GPU. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pp. 47:1–47:9, New York, NY, USA, 2009. ACM.

[17] Aristides G. Requicha. Representations for Rigid Solids: Theory, Methods, and Systems. *ACM Comput. Surv.*, Vol. 12, No. 4, pp. 437–464, December 1980.

[18] A.A.G. Requicha and H.B. Voelcker. Boolean operations in solid modeling: Boundary evaluation and merging algorithms. *Proceedings of the IEEE*,

Vol. 73, No. 1, pp. 30–44, January 1985.

[19] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pp. 463–470, New York, NY, USA, 2003. ACM.

[20] Yutaka Ohtake, Alexander Belyaev, and Marc Alexa. Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. In *Proceedings of the third Eurographics symposium on Geometry processing*, SGP '05, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[21] 金井崇, 大竹豊, 川田弘明, 加瀬究. GPU による点群ベース陰関数曲面の直接的レンダリング. グラフィクスと CAD / Visual Computing 合同シンポジウム, pp. 5–10, 2006.

[22] 柴田章博, 田中覚. 陰関数曲面のブーリアンモデリングとモーフィング. グラフィクスと CAD / Visual Computing 合同シンポジウム, pp. 37–42, 2002.

[23] Zhan Yuan, Yizhou Yu, and Wenping Wang. Object-space multiphase implicit functions. *ACM Trans. Graph.*, Vol. 31, No. 4, pp. 114:1–114:10, July 2012.

[24] Jos Stam and Ryan Schmidt. On the velocity of an implicit surface. *ACM Trans. Graph.*, Vol. 30, No. 3, pp. 21:1–21:7, May 2011.

[25] 松宮雅俊, 竹村治雄, 横矢直和. パーティクルシステムと陰関数曲面を用いた仮想粘土モデリング. 日本バーチャルリアリティ学会大会論文集 = Proceedings of the Virtual Reality Society of Japan annual conference, Vol. 5, pp. 225–228, 2000.

[26] 松宮雅俊. 陰関数曲面とパーティクルシステムを用いた仮想粘土細工による自由形状モデリングに関する研究. 学位論文, 奈良先端科学技術大学院大学,

2002.

[27] Gabriele Lohmann. 3次元画像処理. 株式会社ボーンデジタル, 2009.

[28] J. Andreas Bærentzen. Volume Sculpting: Intuitive, Interactive 3D Shape Modelling. In *IMM VIsionday, 2001*, May 2001.

[29] J. Andreas Bærentzen. Octree-based Volume Sculpting. In *LBHT Proceedings of IEEE Visualization '98*, 1998.

[30] Ralf Böonning and Heinrich Müuller. Interactive sculpturing and visualization of unbounded voxel volumes. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, SMA '02, pp. 212–219, 2002.

[31] Anthony Prior. "On-the-fly" voxelization for 6 degrees-of-freedom haptic virtual sculpting. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, VRCIA '06, pp. 263–270, 2006.

[32] WN Martin and JK Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5, pp. 150–158, 1983.

[33] A. Laurentini. How far 3D shapes can be understood from 2D silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 188–195, 1995.

[34] Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. ł1-Sparse reconstruction of sharp point set surfaces. *ACM Trans. Graph.*, Vol. 29, No. 5, pp. 135:1–135:12, November 2010.

[35] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. GlobFit: consistently fitting primitives by discovering

global relations. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pp. 52:1–52:12, New York, NY, USA, 2011. ACM.

[36] 渡辺賢悟, 宮岡伸一郎. "3D スーラ"：3D 点群情報による点描画ウォークスルーコンテンツ. 芸術科学会論文誌, Vol. 10, No. 3, pp. 192–200, 2010.

[37] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '04, pp. 141–151, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[38] Dan Gerszewski, Haimasree Bhattacharya, and Adam W. Bargteil. A point-based method for animating elastoplastic solids. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pp. 133–138, New York, NY, USA, 2009. ACM.

[39] Patrick Reuter, Ireneusz Tobor, Christophe Schlick, and Sébastien Dedieu. Point-based modelling and rendering using radial basis functions. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '03, pp. 111–118, New York, NY, USA, 2003. ACM.

[40] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3D: an interactive system for point-based surface editing. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pp. 322–329, 2002.

[41] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pp. 641–650, 2003.

[42] Sagi Katz, Ayellet Tal, and Ronen Basri. Direct visibility of point sets. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[43] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pp. 335–342, 2000.

[44] Bart Adams and Philip Dutré. Interactive boolean operations on surfel-bounded solids. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pp. 651–656, 2003.

[45] Anders Adamson and Marc Alexa. Anisotropic point set surfaces. In *Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, AFRIGRAPH '06, pp. 7–13, New York, NY, USA, 2006. ACM.

[46] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[47] Marc Alexa and Anders Adamson. Interpolatory point set surfaces - convexity and hermite data. *ACM Trans. Graph.*, Vol. 28, pp. 20:1–20:10, May 2009.

[48] 鳥谷浩志, 千代倉弘明. 3 次元画像処理 CAD の基礎と応用. 共立出版, 1991.

[49] P.R. Wilson. Euler Formulas and Geometric Modeling. *IEEE Computer Graphics and Applications*, Vol. 5, No. 8, pp. 24–36, August 1985.

[50] M. Mantyla and R. Sulonen. GWB: A Solid Modeler with Euler Operators. *IEEE Computer Graphics and Applications*, Vol. 2, No. 7, pp. 17–31, 1982.

[51] H. Toriya, T. Satoh, K. Ueda, and H. Chiyokura. UNDO and REDO Operations for Solid Modeling. *IEEE Computer Graphics and Applications*, Vol. 6, No. 4, pp. 35–42, April 1986.

[52] 千賀宏樹, U.Ingelstrom, 東正毅, 穂坂衛. 拘束指定によるフィーチャモデリング（第３報）－寸法拘束の拡張とその変更法－. 1999 年度精密工学会秋季大会学術講演会, p. 138, 1999.

[53] Geoffrey I. Webb. Feature based modelling: A methodology for producing coherent, consistent, dynamically changing models of agents' competencies. pp. 117–150, 1996.

[54] Min Tang, Shang-Ching Chou, and Jin-Xiang Dong. Collaborative virtual environment for feature based modeling. In *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, VRCAI '04, pp. 120–126, New York, NY, USA, 2004. ACM.

[55] K.P. Herndon R.C. Zeleznik and J.F. Hughes. SKETCH: An interface for sketching 3D scenes. In *Proceedings of ACM SIGGRAPH '96*, pp. 163–170, 1996.

[56] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3D freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pp. 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[57] Takeo Igarashi and John F. Hughes. A suggestive interface for 3D drawing. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[58] Yuki Mori and Takeo Igarashi. Plushie: an interactive design system for plush toys. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[59] Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. Analytic drawing of 3D scaffolds. *ACM Trans. Graph.*, Vol. 28, No. 5, pp. 1–10, 2009.

[60] Johannes Schmid, Martin Sebastian Senn, Markus Gross, and Robert W. Sumner. OverCoat: an implicit canvas for 3D painting. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pp. 28:1–28:10, New York, NY, USA, 2011. ACM.

[61] Alec Rivers, Frédo Durand, and Takeo Igarashi. 3D modeling with silhouettes. In *ACM SIGGRAPH 2010 papers*, SIGGRAPH '10, pp. 109:1–109:8, New York, NY, USA, 2010. ACM.

[62] 近藤邦雄. 陰関数曲面を用いた集合演算によるスケッチモデリング. 日本図学会, 2006.

[63] Yotam Gingold, Takeo Igarashi, and Denis Zorin. Structured annotations for 2D-to-3D modeling. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, pp. 1–9, New York, NY, USA, 2009. ACM.

[64] Alexis ANDRE, Suguru SAITO, and Masayuki NAKAJIMA. Single-View Sketch Based Surface Modeling. *IEICE transactions on information and systems*, Vol. 92, No. 6, pp. 1304–1311, 2009.

[65] TOYOMITSU SENDA. Reconstruction of Solid from a Set of the Orthographical Three Views : Applications to Many Polyhedral Solids. *Transactions of Information Processing Society of Japan*, Vol. 31, No. 9, pp. 1312–1320, 1990.

[66] 馬場弘行, 藤本直樹, 小堀研一, 久津輪敏郎. フィレット曲面を含む三面図から立体自動生成の一手法. 電子情報通信学会秋季大会講演論文集, Vol. 1994, p. 356, 1994.

[67] 増田宏, 沼尾雅之. 不完全な三面図からの立体モデルの生成. 人工知能学会誌, Vol. 12, No. 6, pp. 928–935, 1997.

[68] Y.Yagi, D.Arita, and R.Taniguchi. Real-Time 3D Shape Modeling from Multi-View Image. In *Proc.8th Joint Workshop -Frontiers of Computer Vision-*, pp. 55–61, 2002.

[69] Thorsten Thormählen and Hans-Peter Seidel. 3D-modeling by ortho-image generation from image sequences. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pp. 86:1–86:5, New York, NY, USA, 2008. ACM.

[70] Mark Finch, John Snyder, and Hugues Hoppe. Freeform vector graphics with controlled thin-plate splines. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, pp. 166:1–166:10, New York, NY, USA, 2011. ACM.

[71] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves: a vector representation for smooth-shaded images. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pp. 92:1–92:8, New York, NY, USA, 2008. ACM.

[72] Yotam Gingold and Denis Zorin. Shading-based surface editing. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pp. 95:1–95:9, New York, NY, USA, 2008. ACM.

[73] Tai-Pang Wu, Chi-Keung Tang, Michael S. Brown, and Heung-Yeung Shum. ShapePalettes: interactive normal transfer via sketching. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[74] 今泉仁美, 伊藤貴之. IGEL -ヒートカッターを模した3次元形状モデリング-. 芸術科学会論文誌, Vol. 8, No. 2, pp. 43–50, 2009.

[75] Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. iWIRES: an analyze-and-edit approach to shape manipulation. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pp. 33:1–33:10, New York, NY, USA, 2009. ACM.

[76] Ronald N. Perry and Sarah F. Frisken. Kizamu: a system for sculpting digital characters. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pp. 47–56, 2001.

[77] Roland Hess. *The Essential Blender: Guide to 3D Creation with the Open Source Suite Blender*. No Starch Press, San Francisco, CA, USA, 2007.

[78] Eric Keller. *Introducing ZBrush*. Wiley, Hoboken, NJ, 2008.

[79] PILGWAY Co. . 3D-Coat. http://3d-coat.com/, accessed August 10, 2012.

[80] Shinji MIZUNO. Proposal and Implementation of Virtual Chinkin. 画像電子学会誌, Vol. 40, No. 1, pp. 132–140, 2011.

[81] Manfred Lau, Akira Ohgawara, Jun Mitani, and Takeo Igarashi. Converting 3D furniture models to fabricatable parts and connectors. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pp. 85:1–85:6, New York, NY, USA, 2011. ACM.

[82] Tom Kelly and Peter Wonka. Interactive architectural modeling with procedural extrusions. *ACM Trans. Graph.*, Vol. 30, No. 2, pp. 14:1–14:15, April 2011.

[83] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. In *ACM*

SIGGRAPH 2011 papers, SIGGRAPH '11, pp. 87:1–87:10, New York, NY, USA, 2011. ACM.

[84] Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. Make it home: automatic optimization of furniture arrangement. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pp. 86:1–86:12, New York, NY, USA, 2011. ACM.

[85] Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. Metropolis procedural modeling. *ACM Trans. Graph.*, Vol. 30, No. 2, pp. 11:1–11:14, April 2011.

[86] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3D mesh segmentation and labeling. In *ACM SIGGRAPH 2010 papers*, SIGGRAPH '10, pp. 102:1–102:12, New York, NY, USA, 2010. ACM.

[87] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J. Mitra. Exploration of continuous variability in collections of 3D shapes. *ACM Trans. Graph.*, Vol. 30, No. 4, pp. 33:1–33:10, July 2011.

[88] Siddhartha Chaudhuri and Vladlen Koltun. Data-driven suggestions for creativity support in 3D modeling. In *ACM SIGGRAPH Asia 2010 papers*, SIGGRAPH ASIA '10, pp. 183:1–183:10, New York, NY, USA, 2010. ACM.

[89] Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. Probabilistic reasoning for assembly-based 3D modeling. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pp. 35:1–35:10, New York, NY, USA, 2011. ACM.

[90] Alexander M. Bronstein, Michael M. Bronstein, Leonidas J. Guibas, and Maks Ovsjanikov. Shape google: Geometric words and expressions for invariant

shape retrieval. *ACM Trans. Graph.*, Vol. 30, No. 1, pp. 1:1–1:20, February 2011.

[91] Jonathan D. Denning, William B. Kerr, and Fabio Pellacini. MeshFlow: interactive visualization of mesh construction sequences. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pp. 66:1–66:8, New York, NY, USA, 2011. ACM.

[92] Fei-Yue Wang. A simple and analytical procedure for calibrating extrinsic camera parameters. *Robotics and Automation, IEEE Transactions*, Vol. 20(1), pp. 121–124, feb 2004.

[93] L.B. Kara, L. Gennari, and T.F. Stahovich. A Sketch-Based Tool for Analyzing Vibratory Mechanical Systems. *Journal of Mechanical Design*, Vol. 130, p. 101101, 2008.

[94] L.B. Kara and K. Shimada. Supporting early styling design of automobiles using sketch-based 3D shape construction. *Computer-Aided Design and Applications*, Vol. 5, No. 6, 2008.

[95] Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis. *OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R) Version 2.* Addison-Wesley Professional, 5th edition, 2005.

[96] Fine Kernel Project. Fine Kernel ToolKit System. http://fktoolkit.sourceforge.jp/, accessed August 10, 2012.

[97] Hsiang-Ting Chen, Li-Yi Wei, and Chun-Fa Chang. Nonlinear revision control for images. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pp. 105:1–105:10, New York, NY, USA, 2011. ACM.

[98] Ryota Takeuchi, Taichi Watanabe, and Soji Yamakawa. Sketch-based Solid Prototype Modeling System with Dual Data Structure of Point-set Surfaces and Voxels. *International Journal of CAD/CAM*, Vol. 11, No. 1, 2011.

発表実績

学術論文

Ryota Takeuchi, Taichi Watanabe, Masanori Kakimoto, Koji Mikami and Kunio Kondo,
Stroke history management for digital sculpting,
The Journal of the Society for Art and Science, Vol. 11, No. 4, pp. 108-117, 2012.

Ryota Takeuchi, Taichi Watanabe and Soji Yamakawa,
Sketch-based Solid Prototype Modeling System with Dual Data Structure of Point-set Surfaces and Voxels,
International Journal of CAD/CAM, Vol. 11, No. 1, 9 pages, 2011.

国際会議 (査読あり)

Ryota Takeuchi, Taichi Watanabe, Koji Mikami and Kunio Kondo,
Proposal of digital sculpting with history management of strokes,
NICOGRAPH International, pp. 76-82, 2012.

Ryota Takeuchi and Taichi Watanabe,
Illustration based sculpture modeling system by point set surface,
ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI), 3 pages, 2009.

国内会議 (査読あり)

竹内亮太, 渡辺大地,
フィーチャーベースドのメタファを加えたスカルプトモデリング手法の提案,

Visual Computing / グラフィクスと CAD 合同シンポジウム, 4 pages, 2011.

国内会議 (査読なし)

竹内亮太, 渡辺大地,
ポイントグラフィクスを用いた切削による形状編集手法,
情報科学技術フォーラム (FIT), 2 pages, 2007.

竹内亮太, 渡辺大地,
陰関数曲面を用いた点群形状モデリング手法の提案,
芸術科学会：第 23 回 NICOGRAPH 論文コンテスト, 6 pages, 2007.

竹内亮太, 渡辺大地,
異なる物質の特性を表現した点群形状モデリング手法の提案,
情報処理学会「グラフィクスと CAD」研究報告 2008-CG-132, 4 pages, 2008.

竹内亮太, 渡辺大地,
フィーチャーベースドのメタファを加えたスカラプトモデリング手法の提案,
画像電子学会：ビジュアルコンピューティングワークショップ 2011 in 加太温泉,
2011.

受賞など

CG-ARTS 協会賞,
東京工科大学における CG コンテンツの制作技術教育の功績により受賞 (2010/2).