SilF: 3D アウトラインによる擬似 3D 表現を用いたスケッチツール

SilF: A Sketching Tool for Cartool-Like Pseudo-3D Illustrations Based on 3D Outlines

米澤 航太*
Kota Yonezawa
東京工業大学
情報理工学研究科
数理・計算科学専攻
Graduate School of Information Science and Engineering, Tokyo Institute of Technology
Kota.Yonezawa@is.titech.ac.jp

高橋 伸
Shin Takahashi
筑波大学
システム情報工学研究科
Systems and Information Engineering, University of Tsukuba
shin.takahashi@acm.org

柴山 悦哉
Etsuya Shibayama
東京工業大学
情報理工学研究科
数理・計算科学専攻
Graduate School of Information Science and Engineering, Tokyo Institute of Technology
etsuya@is.titech.ac.jp

概要

本論文では，マンガ風のイラストレーションのための，3D 空間上での視点変更や回転が可能な，新しい擬似 3D 表現とそのレンダリングアルゴリズムを提案する．マンガ風のイラストレーションの場合，アウトライン（境界線）部分が濃く太く強調される．本表現では，対象となる物体をポリゴン等の面情報を含む完全な 3D 形状として表すのではなく，アウトラインのみで表現する．このアウトラインは，一般の 2D イラストレーションと異なり，3D 空間に，3D 曲線として配置する．これを 3D アウトラインと呼ぶ．3D アウトラインは，まず投影面に投影変換して 2D ベクタイラストレーションと全く同様の形に変換してから描画されるので，描画結果は 2D ベクタイラストレーションと非常に近い風合いになる．3D アウトラインによる表現は，従来の 3D 表現と 2D イラストとの中間的な位置にあるといえる．この限られた 3D 情報しか持たないモデルであっても，イラストレーションを完全な 3D 立体があるかのように 3D 空間内で回転させて見ることが出来，十分な立体感を得ることが可能である．(図 1)

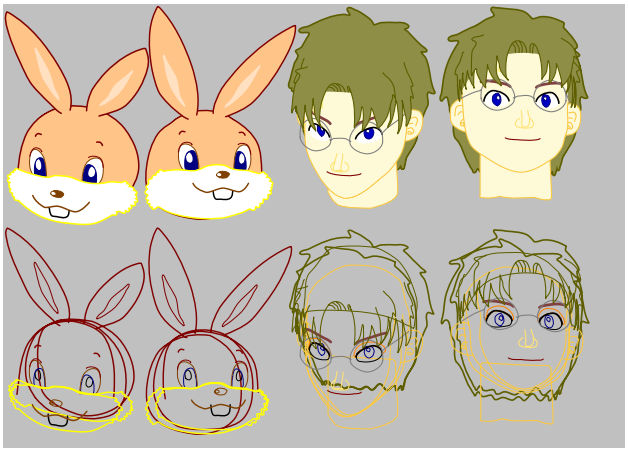

図 1: サンプルモデル

本論文では，この 3D アウトラインモデルをモデリングするためのインタフェースも提案する．本モデルは 2D のイラストレーションに近い性質を持つ事から，インタフェースも 2D イラストレーションと同様なスケッチベースが直感的であると考えられる．我々は，本モデルを一般的な 2D 入力デバイスを用いたスケッチベースのインタフェースで編集し，表示する為のシステム"SilF"を試作した．

はじめに

近年まで，アニメーションは全て大勢のイラストレータによる手作業で描かれていた．それは特殊なスキルが必要で，また大変手間のかかる作業である．
近年，3D 形状を手描きイラスト風にレンダリングする，ノンフォトリアリスティックレンダリングが一般的になりつつある．[5,6] この一形式であるトゥーン・レンダリングによって，3D グラフィックスと従来のセルアニメーションとのシームレスな融合が可能になった．一度 3D モデルを構築すれば，何回もそれを再利用する事が可能になり，作画の手間と時間を大幅に削減する事が可能になった．
さらに，この手法によって，手描きによる対応が難しかったインタラクティブな用途にも対応できるようになった．インタラクティブなシステムでは，ユーザの操作により任意の場所への視点の変更が起こるが，完全な 3D モデルがあれば，どのような角度からの絵も得る事ができる．近年のハードウェアの進歩により，かなりクオリティの高い絵もリアルタイムに表示可能になってきている．
この手法における問題点としては，3D モデル構築の際の難しさが挙げられる．元となる 2D イラストと，それを 3D 化した 3D モデルとの間にはかなり大きなギャップがある．一般的なマンガ風のイラストレーションでは，立体の輪郭線の部分は太く，濃い線で強調されて描かれるが，その内部は少ない色数で塗りつぶされる．つまり，輪郭線の形状が特に重要であるといえる．既存の 2D 編集では，この輪郭線を直接編集することができるため，非常に直感的である．しかし，3D モデルを構築してそれをトゥーン・レンダリングする事によりイラストレーションを作成する場合，強調された輪郭線は 3D モデルをレンダリングした結

* 現所属：ソニー株式会社

果生成されるもので，ユーザが直接編集するものではない．3D モデル作成者は，そのモデルをある視線の位置でレンダリングした時の輪郭線が求める形になるように考慮しながら3D形状を構築する作業が必要になる．このような作業は非常に技術を要する．
マンガ風のイラストレーションの場合，輪郭線とは異なり，輪郭線で囲まれた部分はベタ塗りされてしまうため，輪郭線に近い部分以外の面の形状は必要性が低い．マンガ風のイラストに限定すれば，3D の面を含む完全なモデルなしでも，輪郭線部分の情報のみで，立体を表現できる可能性が考えられる．
本研究ではマンガ風イラストの 3D 空間での視点変更を実現する，新しい擬似 3D 表現を提案する．本手法では，対象となる 3D 形状をアウトラインのみで表現する．このアウトラインは通常の 2D イラストレーションのように 2D 平面上に配置されるのではなく，3D 空間内に配置される．これを，3D アウトラインと呼ぶ．図 1のように，限られた情報しか持たないモデルであっても，十分立体的に見せることが出来る．
このモデルでは，ユーザは従来の 3D モデリング作業とは異なり，2D イラストレーションと同じく，立体の輪郭線に当たる 3D アウトラインを直接編集する事になるので，2D イラストレーションとのギャップも小さい．また，描画結果も 2D ベクタイラストレーションと非常に近い風合いになる．これらの点で，3D アウトラインによる擬似 3D 表現は，2D イラストレーションと従来の 3D 表現との中間に位置すると言える．
本手法の制限として，3D 空間上で正しく立体らしく見える角度に制限があることが挙げられる．3D アウトラインによる擬似表現では，立体の輪郭線の情報しか持っておらず，3D の面の情報はない．そのため，視点を正面から大きくずらすと，3D アウトラインによる輪郭線と，本来の 3D 形状により生成されるべき輪郭線との間に大きなずれが生じ，破綻をきたす．(図 2) しかしながら，この制限内でも十分有用な用途も多々あると考えている．この点については，後に考察する．

図 2: 視点を大きくずらした例

さらに本論文では，3D アウトラインの編集に特化した，一般的な 2D 入力デバイスによるスケッチベースのインタフェースも提案する．
我々は，3D アウトラインモデルの編集及び表示の為のシステム”SilF”(Silhouette Float の略)を実装した．本論文では，まず関連研究について述べた後，3D アウトラインによる擬似 3D 表現の詳細，3D アウトライン編集のためのスケッチベースのインタフェースについて述べる．

関連研究

SKETCH[9]や Teddy[4]は，ユーザの描いた 2D ストロークから 3D モデルを作成するシステムである．本研究と同様，スケッチベースのインタフェースを用いているが，作成するモデルは，面をもった完全な 3D 形状である．
SKETCH や Teddy と異なり，完全な 3D 形状を用いずに 3D のような効果を生み出す研究がいくつか発表されている．Harold[3]や Flat3D[8]は，ユーザのストロークにより 3D 空間上に透明なキャンバスを生成し，その上にユーザが 2D の絵を描く事により，3D ライクな空間を表現する，インタラクティブなシステムである．
Drawing for Illustration and Annotation in 3D[1]は，方向性が本研究に一番近いシステムである．立体の輪郭線は，見る角度により変化する．言い換えれば，ある角度からみた輪郭線は，視点がその角度から離れるにつれて信頼性が低くなると考える事が出来る．このシステムでは，ユーザがストロークを描くと，それをその角度から見たある立体の輪郭線とみなす．輪郭線として雨樋のような形状の立体を生成し，描いたときの角度から見たときは濃く見え，そこから視点が離れるにつれて薄く消えていくように特殊な方法でレンダリングする事によって，完全な3D形状がそこにあるかのように表現する．既存の一般的な 3D シーン内に注釈をつけることを目的とし，また輪郭線を 3D の面をもつ形状として生成する点は本研究と異なるものの，輪郭線のみで立体を表現するという方向性は一致している．また，本システムは，一部このシステムと同様のインタフェースを採用している．
最後に，An Interface for Sketching 3D Curves[2]では，3D 空間内に曲線を描く為のインタフェースが提案されている．ユーザは空中に 2D ストロークをまず描き，次に地面に影のストロークを描く事により，3D 曲線が決定される．本システムのインタフェースでは，このシステムと似たアルゴリズムを使用している．

3D アウトライン

3D アウトラインとは，線の太さや色，内部の色，内部を塗りつぶすかどうか，などの情報を持つ 3D 曲線である．ここで言う内部の色とは，3D 空間に存在するオブジェクトの色ではなく，描画の際，3D 曲線を投影面に投影した結果，形成された 2D 曲線によって囲まれた 2D 領域を塗りつぶす色である．ポリゴンによる 3D 表現のように，3D の面に色を割り当てる考え方とは根本的に異なっている．そのため，3D アウトラインモデルに 3D の”面”は存在しない．

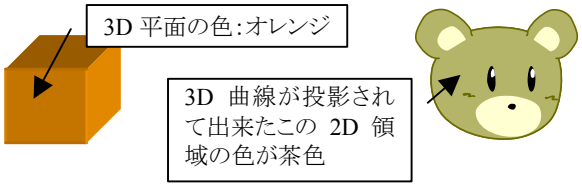

図 3: 既存の 3D 表現と 3D アウトライン表現の違い

3D アウトラインモデルは限られた 3D 情報しか持ってい

ないが，非常に少ない数の 3D アウトラインでも立体を表現する事が可能である．例えば，図 4では，顔の部分はただ一つの 3D アウトラインで構成されている．このように顎の部分を手前に出したような形状にすることで，角度を変化させたときの 3D アウトラインの描画結果が実際の顔の 3D 形状を回転させたときの輪郭に近くなり，まるでそこに顔面があるかのような立体感を得られる．また，目や鼻等のパーツを加える事により，それらが顔面の上に乗っているように見えるので，立体感がより強調される．

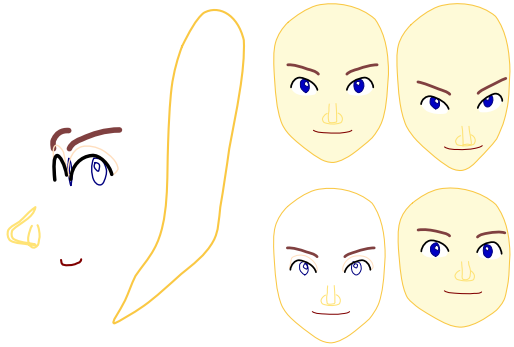

図 4: 簡単な顔の例

本システム"SilF"では，3D アウトラインは 3 次 3D ベジェ曲線として実装している．3D アウトラインの描画手順は，2 つのステップに分けることが出来る．まずそれらを投影面に投影し，通常の 2D ベクタイラストレーションと全く同等の 2D の曲線群に変換する．（ステップ 1）次に，それを一般的な 2D ベクタイラストレーションと同様，2D ベジェ曲線描画関数を用いて描画する．（ステップ 2）
ステップ 1 では，3 次 3D ベジェ曲線である 3D アウトラインを投影面に投影して，3 次 2D ベジェ曲線に変換する．投影には平行透視投影を用いているので，変換は 3D アウトラインのコントロールポイント及びアンカーポイントを，単純な内積計算により 2D 座標に変換する事によって行うことができる．
ステップ 2 では，変換された 2D ベジェ曲線を描画し，もし設定されている場合はその内部を塗りつぶす．”SilF”では，Win32API^{TM}のベジェ曲線描画関数を用いて描画を行っている．

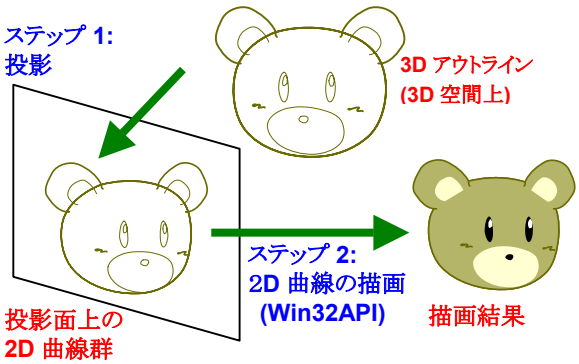

図 5: 3D アウトラインの描画

3D アウトラインによる擬似 3D 表現では，3D 空間での面の情報が存在しない．このため，陰面，陰線消去を一般的な 3D 表現のように自動的に行う事は難しい．
また，前述のように，3D アウトラインによる擬似 3D 表現では，正しく見える視点の角度の範囲に制限がある．一般的に，ある視点から見た時の，2 つの 3D 物体が”重なる”順番，つまり視点からの距離の大小は，視点がそこから大きく離れない限り，変化しない可能性が高い．
このような理由から，本システムでは陰面，陰線消去には特殊な対策は取らず，描画にはペインターズアルゴリズムを採用している．即ち，3D アウトライン同士でどちらが上に描画されるかは，描画のステップ 2 で，どちらが後に描画されるかによる．この 3D アウトラインを”重ねる”順番は，一般的な 2D ベクタドローイングツールと同様，前面へ移動，背面へ移動等のメニューによる指示をユーザが各 3D アウトラインに与える事によって決定する．

複合 3D アウトライン

一般的な 3D 形状では，視点によって輪郭線は変化する．例えば，人の顔では，正面から見た場合は顎のラインが，少し横から見ると頬のラインが輪郭線となる．頬は膨らんでいるため，輪郭線も膨らんだ形状に変化しないと不自然である．このような視線による立体の輪郭線の変化を表現するため，本論文では複合 3D アウトラインを新たに提案する．これは，複数の基本 3D アウトラインにより構成される．

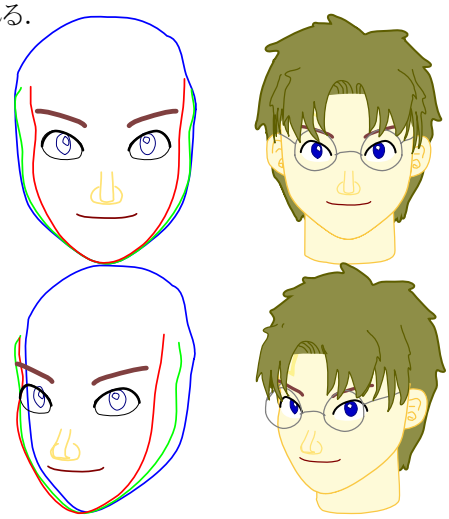

図 6: 複合 3D アウトラインのサンプル
顔の部分が複合 3D アウトラインで構成されている．上行は正面から見た場合である．青の顎の輪郭が描かれ，緑，赤のラインは描画されない． 下行は少し斜めの視点から見た場合である．一部で，頬の輪郭である緑と赤のラインが描画されている．

描画の際，描画のステップ1で 3D アウトラインを構成する基本 3D アウトラインを投影して 2D ベジェ曲線に変換した後，それらをステップ 2 で描画するとき，塗りつぶされる領域を結合させる．つまり，この結合領域内に描かれるはずのアウトラインは描画されない．実際に描画されるのは，その視点から見て一番”外側”に見える 3D アウトラインとなる．これにより視点の変化による立体の輪郭線の

変化を擬似的に表現することができる．図 6は，顔の部分を 3 つの基本 3D アウトラインからなる複合 3D アウトラインで構成した例である．この場合，正面から見たときの輪郭と斜めから見たときの輪郭を複合 3D アウトライン化することにより，顔の複雑な形状を表現している．
複合 3D アウトラインのレンダリングステップ 2 で，複数の 2D ベジェ曲線で塗りつぶす領域を結合させるには，正確には2Dベジェ曲線同士の交点を求める処理が必要となるが，本システムでは，次のような手法を用いている．まず曲線のみ描画し，その後内部を塗りつぶす．(図 7) このような方法により，複合 3D アウトラインを高速に描画出来る．

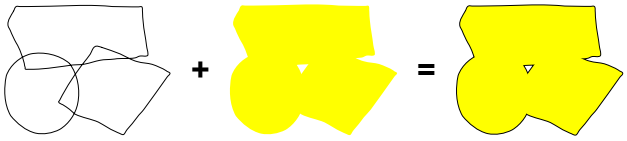

図 7: 塗りつぶす領域の結合

スケッチベースインタフェース

本論文で提案する擬似 3D 表現は，2D イラストレーションに近い性質を持つため，インタフェースも 2D に近い，スケッチベースのインタフェースが自然である．SilF では，マウス，ペンタブレット等の一般的な 2D 入力デバイスの使用を想定している．ただ，これらのデバイスでは 3 次元座標の入力が出来ない為，ユーザの描いた一回のパスで任意の 3D アウトラインを決定できない．そこで，まずユーザの操作により，求める形に近い 3D アウトラインを生成し，そしてそれを修正するという，2 ステップで任意の 3D アウトラインを編集するインタフェースを提案する．

(1) 生成のインタフェース

まず 3D アウトラインの生成ステップでは，ユーザにより描かれた2D のパスを 3D 空間上の平面上に配置することで，3D アウトラインを生成する．基本的には，その平面は現在の視線に垂直な面となる．ここで，パスの始点・終点が既存の 3D アウトラインの線上に位置するように描いた場合，つまり，3D アウトラインが描画のステップ 1 で投影変換された 2D 曲線の上に，始点や終点が描かれた場合，システムは，3D アウトラインを配置する際の 3D 座標を決定する為のユーザによるジェスチャとみなす．この場合，ストロークを3D 空間に配置する際，新しい 3D アウトラインの始点，終点がその既存の 3D アウトライン上に位置するように投影先の平面を調整する．例えば，始点，終点とも既存の 3D アウトライン上に描かれた場合，生成される 3D アウトラインはその 2 点間に橋をかけたような形に生成される．(図 8) ストロークの始点，終点により 3D 空間上の配置を決定する同様のインタフェースは，[1,3] にも見られる．
この他，描いたストロークを 3D 空間に配置する 3 次元空間の面をユーザが決定する事が出来る，ガイドと呼ぶシステムも実装している．このガイドを用いている場合，全ての新しい 3D アウトラインは，このガイドの面の上に配置される．(図 9)人の顔の例では目や眉毛のように，一般の 3D グラフィックスでは 3D 面上に 2D 画像としてマッピングされるような部分を描くのに便利である．
なお，本システムでユーザのストロークから 3D ベジェ曲線を生成する処理は，[7]を 3D 空間に拡張したものを用いている．

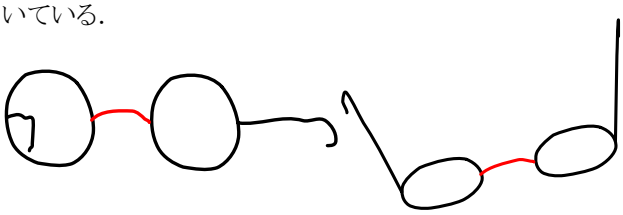

図 8: 3D アウトライン生成の例
左の視点から，ストロークの始点，終点がメガネの縁の上に来るように，メガネのブリッジを描くと，ブリッジの 3D アウトラインも縁の間をつなぐ形に生成される．(右)

図 9: ガイドの使用例
顔のパーツはガイドの上に描かれている．

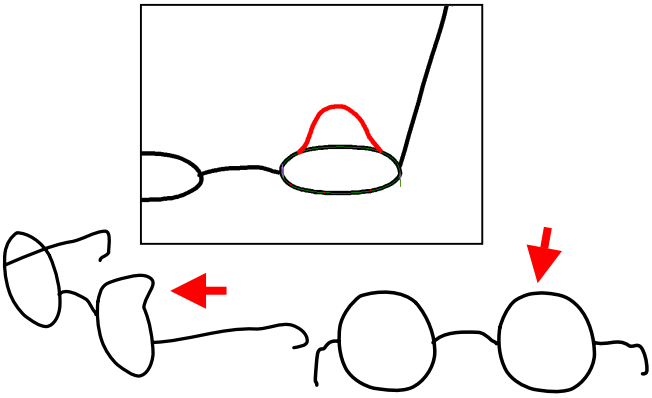

図 10: 奥行き方向への変形例

(2) 修正のインタフェース

生成のステップでは，生成される 3D アウトラインは平面上に配置される．つまりこの時点では，生成する時の視点から見て奥行き方向の座標は自動的に決定されている．修正のステップでは，生成のステップで自由に編集する事が出来なかった奥行き方向へ任意の変形を可能にする．図 10は，奥行き方向への変形例である．ユーザは 3D アウトラインを修正する事を示す為，対象の 3D アウトラインを選択状態にする．これは，3D アウトラインの上でクリックする事により行う．次に上からの視点に変更し，修正ストロークを描く．(上) すると，メガネの縁はメガネの縁の正面から見て奥行き方向に変形される．(下

左）奥行き方向のみに変形された事は，正面からの見た目が変化していない事からも分かる．（下右）
本システムでは，ユーザが修正ストロークを描く時の視点を，ユーザによるジェスチャとして用いている．生成した時と近い視点から描いた場合と，図 10の例のように，3D アウトラインを生成した時の視点と大きくずれた角度から修正ストロークを描いた場合とで，システムが行う修正は異なる．システムは 3D アウトラインが生成された時の視点の方向を各 3D アウトラインごとに記憶している．これを，3D アウトラインの正面と呼ぶ．正面近い角度から修正した場合，修正部分のストロークは生成ステップと同様，ストロークの始点と終点から配置する面が決定され，既存の部分と置き換えられる．正面から視点を大きくずらすと，システムは奥行き方向のみに修正方向の制約をかけて修正を行う．

(3) その他のインタフェース

この他に，既存の 3D アウトラインをなぞるようにストロークが描かれた場合，システムはその 3D アウトラインのなぞられた部分のスムージングを行う．
また，スケッチによる編集を行うモードの他に，拡大，縮小，移動，回転，反転等のアフィン変換を行うモードも用意してある．既存のシステムと同様，3D アウトラインのカット・コピー・ペーストや，複数回のアンドゥ・リドゥもサポートしている．

実装と考察

本システム SilF は，Visual C++ 6.0 Professional™ を用いて Windows™ アプリケーションとして実装されている．

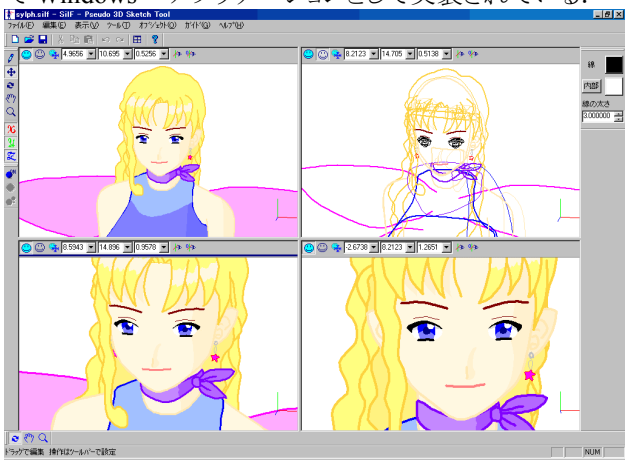

図 11: SilF 概観

実際に SilF を使用して，本論文に示した様々なサンプルを作成した．本論文で示したサンプルでは，正しく見える視線の角度の範囲はサンプルによってかなり差はあるものの，正面から大体 20 度ほどであった．一般的なポリゴンなどによる 3D グラフィックスと比較するとかなり範囲は狭くなるが，エージェントキャラクタのように，あまり正面から大きく回転させる必要のない用途では，十分実用的であると考えられる．もし後ろから見た場面が必要になった場合，前から見た場合のモデルとは別に，後ろから見たモデルも作っておき，場面によって切り替えれば，十分対応できると考えられる．
この他，3D アウトラインを投影面に投影した時点で既存の 2D ベクタイラストレーションと同様に扱えるようになるので，近年よく使われるようになってきたベクタアニメーション用や，印刷用のイラストレーションの 3D 化による再利用性の向上も期待出来る．
本システム SilF を用いたサンプルの作成には，図 12 に示した妖精程の例の場合，慣れれば 2-3 時間程度で作成可能であろう．ただ，3D アウトラインをどのような形状にすれば 3D 形状のように見えるかについては，直感的には分かりにくい為，試行錯誤しながら調整しなければならず，もっと多くの時間がかかる可能性がある．この点については，ユーザテストを行い，考察していく必要があると考えている．

まとめと将来課題

マンガ風イラストレーションにおける，3D アウトラインによる新しい擬似 3D 表現と，その編集のためのインタフェースを提案した．
今後の課題として，ユーザインタフェースの改善，一般的な 3D 表現との比較の為のユーザテストの実施が挙げられる．また，考察で述べたエージェントキャラクタに必要な程度のアニメーション機能についても，検討していきたいと考えている．

参考文献

1. David Bourguignon, Marie-Paule Cani, and George Drettakis. Drawing for Illustration and Annotation in 3D. *EUROGRAPHICS 2001 Conference Proceedings*, pages 114-122, 2001.
2. J.M. Cohen, L. Markosian, R.C.Zeleznik, and J.F. Hughes. An Interface for Sketching 3D Curves. *1999 Symposium on Interactive 3D Graphics Proceedings*, pages 17-21, 1999.
3. J.M. Cohen, J.F. Hughes, and R.C. Zeleznik. Harold: A world made of drawings. *Non Photorealistic Animation and Rendering Conference Proceedings*, pages 83-90, 2000.
4. Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A Sketching Interface for 3D Freeform Design. *SIGGRAPH 99 Conference Proceedings*, pages 409-416, 1999.
5. Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, and Lubomir D. Bourdev. Real-Time Nonphotorealistic Rendering. *SIGGRAPH 97*, pages 59-66, 1997.
6. Barbara J. Meier. Painterly rendering for animation. *SIGGRAPH 96*, pages 477-484, 1996.
7. P.J. Schneider. An algorithm for automatically fitting digitized curves. *ACM Graphics Gems*, pages 612-626, 1990.
8. Hiroaki Tobita and Jun Rekimoto. Flat3D: Drawing a 3D Scene with Freehand Sketching. *Interaction 2001 Proceedings*, pages 105-112, 2001.
9. R.C. Zeleznik, K.P. Herndon, and J.F. Hughes. SKETCH: An interface for sketching 3D scenes. *SIGGRAPH 96 Conference Proceedings*, pages 163-170, 1996.

図 12: やや複雑なモデル例