筑波大学 情報学群 情報メディア創成学類

卒業研究論文

単一景観画像からの
半自動三次元形状モデリングに関する研究

菊池 敬済

指導教員　三谷 純、金森 由博

2016年1月

概要

近年の研究で、単一の景観画像から三次元シーンモデルを生成する技術が開発されてきた。その中の多くの手法は画像中の地平線や消失線を指定することで三次元シーンモデルを生成している。そのため、物体を近くで撮影した写真のように地面が写っていない画像ではこれらの手法で三次元シーンモデルを生成することは難しい。このような画像の三次元シーンを生成するシステムとして、Iizuka らの疎なストローク入力による三次元シーンモデル生成手法があるが、一つのストローク上には同じ奥行きが与えられるため、その部分が画像に対し水平になってしまうという問題がある。本研究では、従来のブラシストロークによる奥行きの指定に加え、ストローク上に形状変形のための制御ハンドルを設置し操作することで、ストローク上の奥行きを変化させられる手法を提案する。これによって一つのストローク上でも奥行きに変化をもたせることができるため、詳細なモデルを生成できる。

目 次

第 1 章　序論 **1**

第 2 章　関連研究 **4**

2.1　手動での境界線の指定による三次元シーンモデルの生成に関する研究 4

2.2　自動での境界線指定による三次元形状の生成に関する研究 4

2.3　疎なデプスの入力による三次元シーンモデルの生成に関する研究 5

第 3 章　提案手法 **8**

3.1　提案システムの流れ 8

3.2　ストロークの入力 8

3.3　デプスマップの生成手法 10

3.4　制御ハンドルを用いたデプス変更処理 12

3.4.1　ストローク操作のためのストロークの構造 12

3.4.2　ストローク移動時の各頂点の動き 16

3.4.3　Sorkine らの手法 [8] によるストローク頂点の移動 16

3.5　背景領域の生成 20

3.6　モデルの修正操作 21

3.6.1　手動での不連続箇所の追加 21

3.6.2　フリーハンド入力によるモデルのデプスの平滑化 22

第 4 章　結果 **23**

4.1　実装したシステム 23

4.1.1　実装言語と作成したユーザインタフェース 23

4.1.2　計算時間の計測 24

4.2　既存手法との比較 24

4.3　生成したモデル群 26

第 5 章　まとめと今後の展望 **28**

5.1　まとめ 28

5.2　今後の展望 28

謝辞 **29**

参考文献 **30**

図 目 次

1.1 境界線の推定による三次元シーンモデル生成の例 2
1.2 地面との境界線が写っていない画像の例 2
1.3 ブラシストロークのみによる三次元シーンモデル生成の例 3

2.1 Horry らの手法 [3] による背景領域の三次元化についての解説 5
2.2 Iizuka らの手法 [4] のワークフロー図 6
2.3 Hoiem らの手法 [5] 6
2.4 Iizuka らの手法 [1] 7

3.1 提案手法の流れ 9
3.2 多重解像度画像を適用したデプスマップの比較 11
3.3 式 3.5 を適用したデプスマップの比較 13
3.4 制御ハンドルを用いたデプス変更の流れ 14
3.5 ストロークの構造イメージ図 15
3.6 二つの制御ハンドルによる各頂点の移動イメージ 17
3.7 三つの制御ハンドルによる各頂点の移動イメージ 18
3.8 Sorkine らの手法 [8] によるメッシュ変形の例 19
3.9 背景領域処理をしたモデルの比較 21
3.10 上手く不連続箇所として検出されなかった例 21
3.11 不連続箇所の指定のイメージ図 22
3.12 平滑化の比較 22

4.1 実装したシステムのユーザインタフェース 23
4.2 Iizuka らの手法 [1] との比較 25
4.3 Iizuka らの手法 [1] を用いて製作した斜めの面を持つモデルとの比較 26
4.4 本手法によって作成したモデル 27

第1章　序論

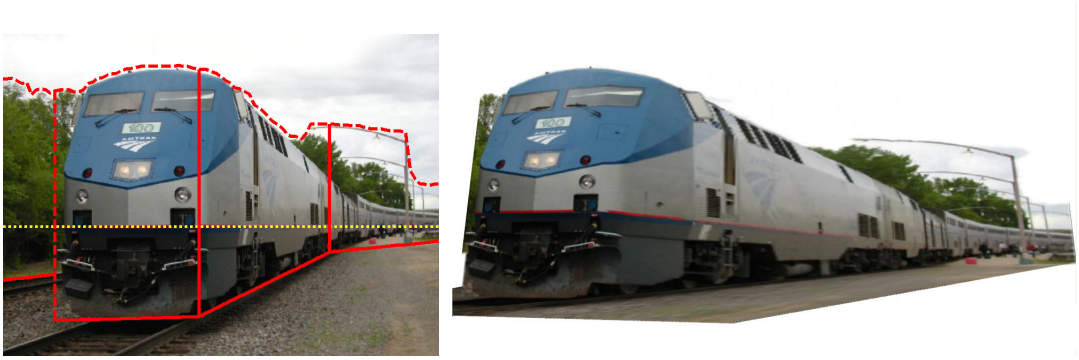
近年では、デジタルカメラの普及により、デジタル写真は多くの人が利用するようになっている。さらに、カメラ付き携帯電話やスマートフォンが普及したことによって、多くの人は自分で撮影した写真をブログやソーシャル・ネットワーキング・サービスに投稿し、世界中の人と共有できるようになった。カメラの補正機能の質の向上や、解像度の向上もあり、多くの人が質の高い写真を見る機会が多くなった。一方で 3DCG も、3D 映画の公開や Google マップのストリートビュー [2] といったサービスの普及により身近に触れる機会が多くなっているが、一般の人が一からリアルな 3D モデルを作成するにはモデリングに関する専門的な知識が多く必要となり、高いハードルとなっている。そのため、誰でも効率的に 3D モデルを製作する手法として画像ベースのモデリング手法が提案されている。この手法は、画像から三次元的な構造を推定し、テクスチャとして画像を貼り付けることでリアリティのある三次元シーンモデルを構築できる。その中でも一枚の画像から三次元シーンモデルを生成する手法は、奥行きや前後関係といった情報が失われているためそれを推測しなければならないが、その点を解決できれば、多くの人が 3DCG を活用することができるようになる。

一枚の画像から三次元形状を復元する研究は、これまで多く行われている。既存研究では図 1.1 のような、画像中の地平線や、地面と物体との境界線を決定し、地面の領域に対し物体の領域を垂直に配置することによって三次元シーンモデルを生成する手法 [3][4][5] が存在する。これらの手法は自動もしくはユーザによる簡単な手動の操作によって三次元シーンモデルを生成できるが、制約条件として、画像中に地平線や物体との境界線が映り込んでいる必要がある。よって、物体を近くから撮影した図 1.2 のような、地面との境界線が写っていない写真については、良好な三次元シーンモデルを生成することは難しい。

地面との境界線が写っていない画像から三次元シーンモデルを生成する手法として、デプスをブラシストロークによって入力し、画像全体に伝搬することでデプスマップを生成する手法が考えられている。ここでのブラシストロークとは、図 1.3(a) のような、画像に対して行われるブラシ入力のことである。Iizuka らの疎なデプスの入力による三次元シーンモデル生成に関する研究 [1] では、ユーザがブラシストロークによって画像の奥行きを入力し、その入力を伝搬させることによってデプスマップを生成する。この手法の問題点は、一つのストローク上では同じ奥行きが与えられるため、図 1.3(b) のように画像に対して平行な部分ができてしまう。また、図 1.3(d) のように段階的な変化を与えようとすると図 1.3(c) のように入力が増えてしまうという問題がある。

本研究では、ブラシストロークによるデプスの入力に加え、ストローク上に形状変形のための制御ハンドルを設置し操作することで、一つのストローク上で奥行きを変化させられる

手法を提案する。これによって一つのストローク上で奥行きの変化を持たせることができるため、詳細なモデルの生成が可能になる。

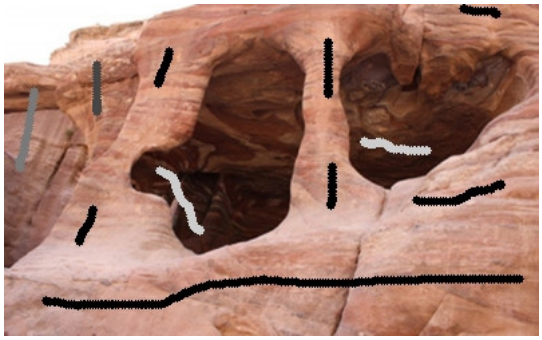


(a) 境界線の推定　　(b) (a) から生成されたモデル

図 1.1: 境界線の推定による三次元シーンモデル生成の例。
出典: Automatic Photo Pop-up[5]

図 1.2: 地面との境界線が写っていない画像の例。
左の画像の出典: ”Colours of Petra”(https://www.flickr.com/photos/adjourned/4700417347), Magnus Franklin [1]
右の画像の出典: 写真素材ぱくたそ (https://www.pakutaso.com/20150928260post-6059.html)

[1]licensed under CC BY 2.0 (http://creativecommons.org/licenses/by/2.0/)



(a) ユーザ入力

(b) (a) に対する出力

(c) 下部が斜めになるようなユーザ入力

(d) (c) に対する出力

図 1.3: ブラシストロークのみによる三次元シーンモデル生成の例。プログラムは Iizuka らの手法 [1] によるもの。(a) の入力では、一つのストローク上で同じ奥行きが与えられるため、(b) のように投影面に対し平行な面ができてしまう。また、(c) のように、斜めな面になるようにモデルを作ろうとするとデプスの入力回数が増えてしまう。

第2章　関連研究

この章では、本研究に関連する、一枚の画像を入力として三次元シーンモデルを復元する既存研究を紹介する。はじめに手動で境界線を指定することで三次元シーンモデルを生成する手法を紹介し、次に自動での境界線の指定によって三次元シーンモデルを生成する手法を紹介する。最後に、疎なデプスの入力による三次元シーンモデルの生成に関する手法を紹介する。

2.1　手動での境界線の指定による三次元シーンモデルの生成に関する研究

手動での境界線の指定による三次元シーンモデルの生成に関する研究は、これまでに多く行われてきた。Horry らは Tour into the picture と呼ばれる手法 [3] を提案した。この手法は、一枚の画像を入力とし、図 2.1 のように消失点とそこから放射状に伸びる spaidery mesh と呼ばれる境界線を指定して、「天井」、「床」、「左壁」、「右壁」、「後壁」の 5 つの領域に分割することでポリゴンモデルを生成して三次元シーンモデルを生成できる。それに加え、複数の前景物を 3D シーン上に指定することができる。この手法では、簡単な手動入力でモデルを生成できるが、消失点が一つの画像にのみ適用できるという制約があるため、二つ以上存在する画像や消失点が存在しない画像は正しく三次元形状が復元できない。

Iizuka らは、手動で地面と物体との境界線を指定することで、地面と複数の壁を生成し、三次元シーンモデルを生成する手法 [4] を提案した。また、全景物を塗りつぶして指定することで前景の遮蔽物を抽出しシーンに配置することで自然なシーンを生成する。モデル生成の流れを図 2.2 に示す。こちらの手法では、消失点が 2 つ以上存在しても三次元シーンモデルを生成することができる。

2.2　自動での境界線指定による三次元形状の生成に関する研究

Hoiem らは一枚の画像から、機械学習によって画像を地面、物体、空の 3 つにラベル付けし、地面との境界を推定することで、三次元シーンモデルを生成する手法 [5] を提案した。入力からシーンモデル生成までの図を図 2.3 に示す。この手法では、機械学習による推定であるため、最初のラベル付けに失敗すると正しいシーンモデルが生成できない。

また、2.1 節の手法と共通して言えることとして、、画像から境界線を指定して三次元形状を生成する手法では、地面との境界が写っている画像でないと正しい三次元シーンモデルを

生成することは難しい。そのため、物体をアップで写したような画像では別の手法を用いる必要がある。

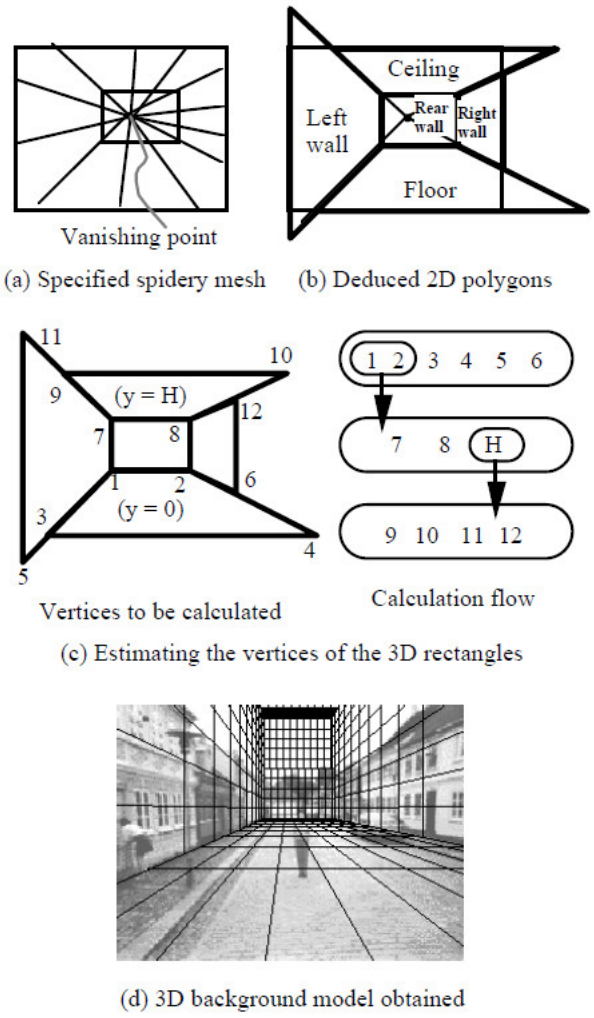


図 2.1: Horry らの手法 [3] による背景領域の三次元化についての解説。ユーザは spaidery mesh と呼ばれる線の集合を指定し (a)、5 つに分割した後 (b)、各頂点を三次元領域に変形し (c)、三次元領域を構成する (d)。

出典: Tour into the picture[3]

2.3 疎なデプスの入力による三次元シーンモデルの生成に関する研究

物体をアップで写した画像の三次元シーンモデルを生成する手法として、画像に対しブラシストロークを入力することでデプスを指定し、それを画像全体に伝搬させてデプスマップを生成するする手法が多くとられている。

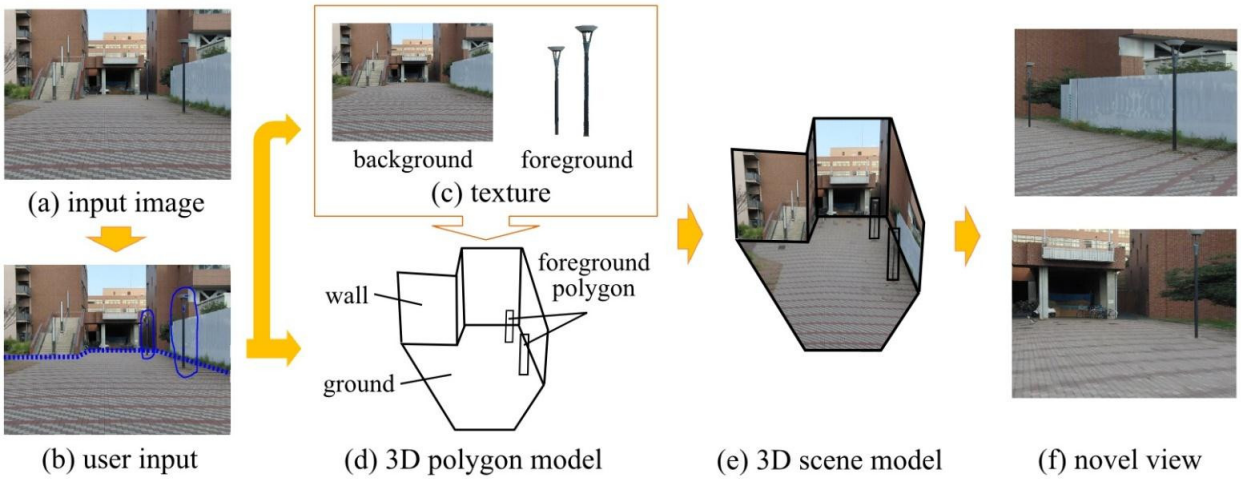
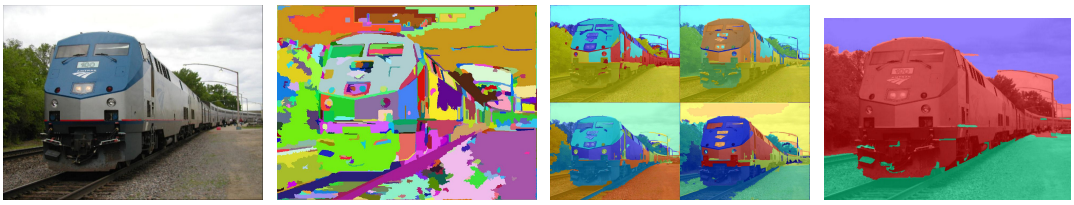


図 2.2: Iizuka らの手法 [4] のワークフロー図。
出典: Efficiently Modeling 3D Scenes from a Single Image[4]



(a) 入力画像　(b) スーパーピクセル　(c) constellations　(d) ラベル付け

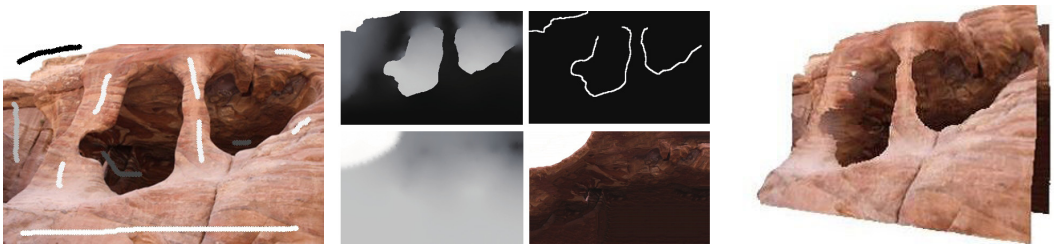
(e) 三次元モデル

図 2.3: Hoiem らの手法 [5]。入力画像 (a) をスーパーピクセルに分割し (b)、統計に基づいて constellation と呼ばれる大きなまとまりを複数生成する (c)。生成した constellation をラベル付けし、最終的なラベルを決定することで (d) 境界線を推測し、三次元シーンモデルを生成する (e)。

出典: Automatic Photo Pop-up[5]

Wang らはブラシストロークによってデプスを入力することで、一枚の画像からアナグリフを生成する StereoBrush と呼ばれる手法 [6] を提案した。この手法では、不連続な部分を保存しつつ平滑化することで物体ごとになめらかなデプスマップを生成する方法を示している。しかしながら、デプスの伝搬をピクセルベースで計算しているため、一本のストロークでデプスが伝搬しにくいという問題がある。

Iizuka らは、ブラシストロークによるデプス入力で一枚の画像から三次元シーンモデルを生成することに加え、図 2.4 で示すように、不連続箇所となる箇所を抽出し切れ込みを入れ背景部分と前景部分の二層にモデルを分けることでリアルな三次元シーンモデルを生成する手法 [1] を提案した。この手法では、スーパーピクセルでデプス伝搬の最適化計算を行っているため、ピクセルベースに比べデプスが伝搬しやすい利点がある。



(a) ユーザ入力　(b) システムの出力　(c) 最終的な三次元モデル

図 2.4: Iizuka らの手法 [1]。ユーザ入力 (a) に対しデプスマップ ((b) 左上)、不連続箇所 ((b) 右上)、背景のデプスマップ ((b) 左下)、背景のテクスチャ((b) 右下) を計算し、前景と背景に分かれた三次元シーンモデル (c) を出力する。
出典: Efficient Depth Propagation for Constructing a Layered Depth Image from a Single Image[1]

これらの手法の問題は、一つのストロークに対して単一のデプスしか割り当てられないため、モデルを生成したときに画像の投影面に対し平行な部分ができてしまい、モデルに不自然な箇所ができてしまう点である。

第3章　提案手法

本章では、提案する三次元シーンモデル生成手法について述べる。はじめに提案手法の流れについて説明する。次に、ストローク入力、デプスマップの生成処理、制御ハンドルを用いたモデル変形処理、背景生成処理と手動でのモデル修正処理についてそれぞれ説明する。

3.1　提案システムの流れ

提案手法では、図 3.1 のような流れで三次元シーンモデル生成を行う。まず、ユーザは三次元シーンモデルを生成したい画像を入力する (図 3.1(a))。次にユーザは、入力した画像に対し以下のどれかの処理を行う。

- ブラシストロークによってデプスを指定する (図 3.1(b))。
- 入力されたブラシストロークに対し制御ハンドルを追加し、追加した制御ハンドルを動かしてモデルの変形操作を行う (図 3.1(c))。
- モデルの修正操作を行う (図 3.1(d))。
- 背景生成の指示を行う (図 3.1(e))。

その後、システムでは与えられた入力から、デプスマップの生成と不連続箇所の抽出、背景生成の指示がされた場合は背景領域を生成する。これを元に三次元シーンモデルを出力し、ユーザはその出力結果に対して、上記のユーザ操作を繰り返すことで、対話的にモデルの生成を行う。

3.2　ストロークの入力

ブラシストロークによるデプスの指定では、グレイスケールのストロークによってデプスを指定する。本手法では、黒に近いほど画像手前側に、白に近いほど画像奥側にデプスが指定される。ブラシストロークが入力されると、最適化計算によってデプスマップが生成され、それに沿って画像に奥行きが付与される。デプスマップの計算方法については、3.4.3 節で説明する。

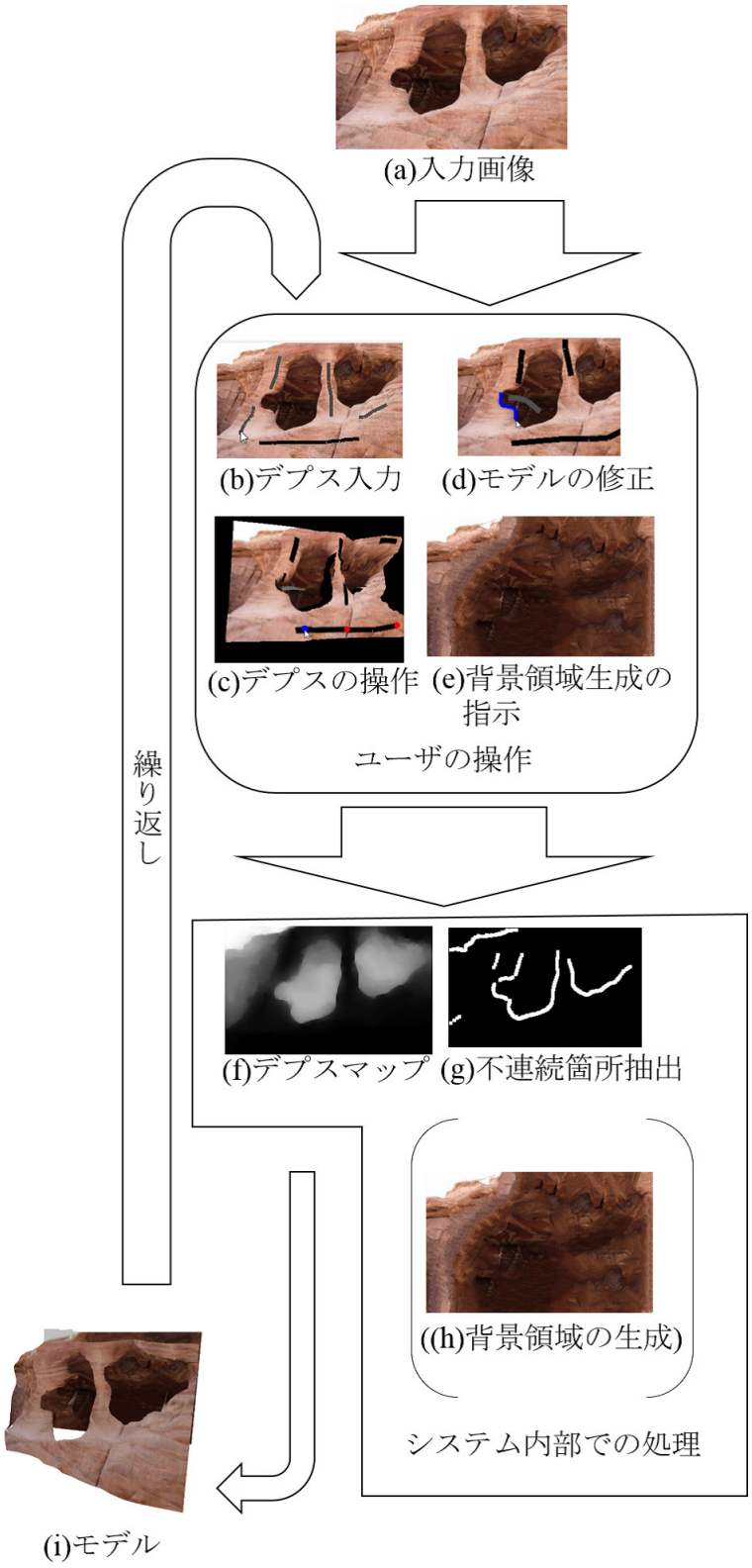

(a)入力画像
(b)デプス入力
(d)モデルの修正
(c)デプスの操作
(e)背景領域生成の指示
ユーザの操作
繰り返し
(f)デプスマップ
(g)不連続箇所抽出
((h)背景領域の生成)
システム内部での処理
(i)モデル

図 3.1: 提案手法の流れ

3.3 デプスマップの生成手法

デプスマップの計算は、制御ハンドルによる変形を考慮してピクセルベースでの最適化計算による手法を用いる。デプスマップの最適化計算は、Wang らの手法 [6] で用いられている以下の式を使用している。

$$\arg\min_{U} \sum_{i \notin M} \left\| U_i - \sum_{q \in N(i)} w_{iq} U_q \right\|^2 + \sum_{i \in M} \|U_i - M_i\|^2 \tag{3.1}$$

ここで、U は最終的なデプスマップ、$N(i)$ はピクセル i の近傍ピクセルの集合、w_{iq} は近傍ピクセル q のピクセル i に対する重み係数、M はユーザによるブラシストロークのマップである。重み係数 w_{iq} は、Levin らの画像に対しブラシストロークによって自然な色を付与する研究 [7] から以下の式によって係数を計算している。

$$w_{iq} \propto \exp\left(-\frac{(Y_i - Y_q)^2}{2\delta_q^2}\right) \tag{3.2}$$

ここで、Y_i は入力画像の RGB 値、δ_q^2 はピクセル i の近傍の分散である。係数を使用する際は、$\sum_{q \in N(i)} w_{iq} = 1$ となるように正規化されて使用される。

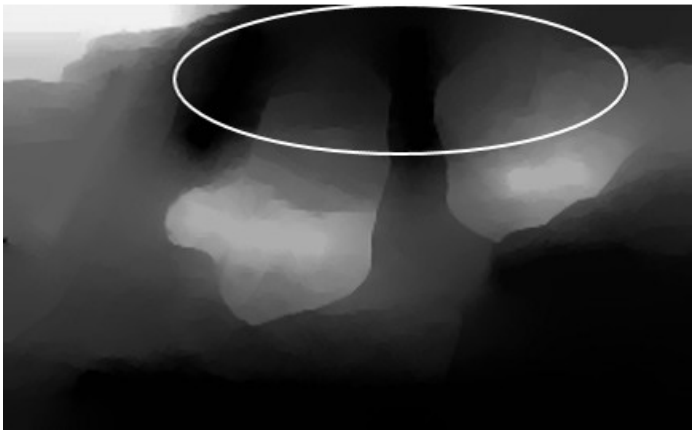
デプスの伝搬手法において、ピクセルベースでの手法では Iizuka らの手法 [1] のようなスーパーピクセルベースでの手法に比べ、デプスが伝搬しにくいという問題点がある。本研究では、Levin らの手法 [7] のような多重解像度画像を使用し改善を図った。解像度の粗い画像から式 3.1 を適用し、生成されたデプスマップを一段階高い解像度へ初期値として適用する。解像度の高い画像への適用方法としては、閾値 ϵ を決め、解像度が高い画像について $\left|U_i - \sum_{q \in N(i)} w_{iq} U_q\right|$ が閾値以下であれば以前のデプスマップの値を引き継ぐ、といった方法をとっている。本手法では、$\epsilon = 3.0$ としている。

図 3.2 に多重解像度画像を利用して生成したデプスマップと利用せずに生成したデプスマップの比較を示す。図中の白丸で囲ってある部分に注目すると、多重解像度画像を利用しないデプスマップに比べ、多重解像度画像を利用したデプスマップの方が、入力したデプスが伝搬していることがわかる。

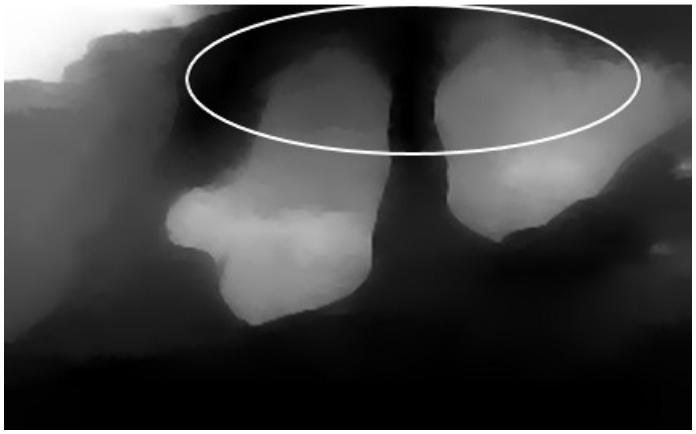
また、3.5 節で述べる背景の生成時に、Wang らの手法 [6] で用いられている、画像の顕著性、エッジ情報を元にデプスマップに重みづけを行い、画像の不連続部分を保持しつつデプスマップを平滑化する処理を行う。これにより、式 3.1 だけの結果よりも、エッジ部分は強調され、それ以外は平滑化されたデプスマップが生成できる。Wang らの論文 [6] では、平滑化項に以下の式を使用している。

$$E_s = \sum_i S_i \left(\sum_{q \in N(i)} \delta_{iq} \left\| (\Omega_q - \Omega_i) - \Delta_x(i, q) \right\|^2 \right) \tag{3.3}$$

(a) 入力



(b) 多重解像度画像を利用しないで生成したデプスマップ



(c) 多重解像度画像を利用して生成したデプスマップ

図 3.2: 多重解像度画像を適用したデプスマップの比較

ここで、S_i は画像の顕著性、$\delta_i q$ はピクセル i と q がエッジに属している場合ゼロに近い値、そうでない場合 1 となる変数、Ω はアナグリフの視差画像、$\Delta_x(i,q)$ はピクセル i と q の x 方向の距離である。本研究はアナグリフの生成でなく三次元シーンモデルの生成が目的なので、以下のような式となる。

$$E_s = \sum_i S_i \left(\sum_{q \in N(i)} \delta_{iq} \left\| (U_q - U_i) \right\|^2 \right) \tag{3.4}$$

式 3.4 と式 3.1 を Wang らの論文 [6] の式を元に最適化の式を用いると、以下のような式となる。

$$\arg\min_U \left\| C_i(U_i - U_i') + w_s E_s \right\|^2 \tag{3.5}$$

C_i は入力されたブラシストロークの距離変換、U' は式 3.1 によって計算されたデプスマップ、w_s はユーザが決められる値で、本手法では Wang らの手法 [6] と同様に $w_s = 400$ で計算を行っている。

図 3.3 に式 3.5 を適用したデプスマップの比較を示す。図 3.3(a) に比べ図 3.3(b) のほうがデプスが滑らかになり、エッジもくっきりとしていることがわかる。しかし、式 3.5 は計算時間が大きく、デプスマップが変更されるたびに計算を行うとインタラクティブな操作ができないため、背景出力時に計算を行っている。

3.4 制御ハンドルを用いたデプス変更処理

図 3.4 に、本手法での制御ハンドルを用いたデプス変更の流れを示す。本手法ではデプス入力に用いるストロークは折れ線で表されており、図 3.4(a) のように、制御ハンドルを折れ線の頂点に配置する。本手法のストロークの構造については、3.4.1 節で説明する。

その後、図 3.4(b) から図 3.4(c) のように、配置された制御ハンドルを動かすことでデプスの変更を行うことができる。この時、一つのストローク上に制御ハンドルが二つ以上設置されている場合、動かしている制御ハンドル以外の制御ハンドルの奥行きは固定され、それ以外の制御ハンドルを配置できる頂点の奥行きは、ストローク上の制御ハンドル間の奥行きを補完するように変化する。この時のストロークの頂点の動きの詳細について、3.5(b) 節で説明する。

本手法では、制御ハンドルの移動による各頂点のデプスの計算を、Sorkine らの手法 [8] を利用して行っている。この手法について、3.4.3 節で説明する。

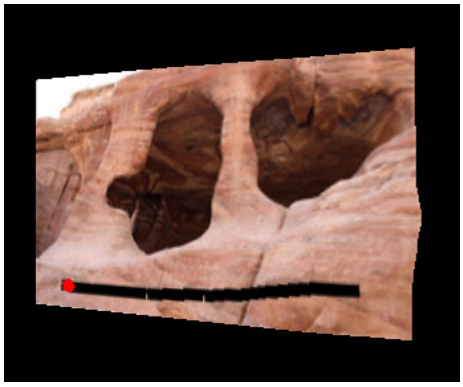
3.4.1 ストローク操作のためのストロークの構造

図 3.5 に、提案手法でのストロークの構成について示す。ストロークは折れ線で構成されており、図 3.5(a) 中の緑丸で示されている折れ線の頂点情報を保持している。ユーザは、その

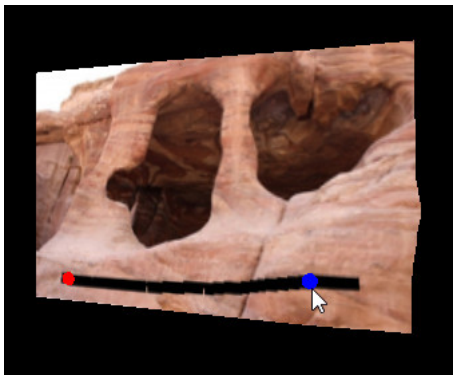
(a) 式 3.5 を適用しないデプスマップ

(b) 式 3.5 を適用したデプスマップ

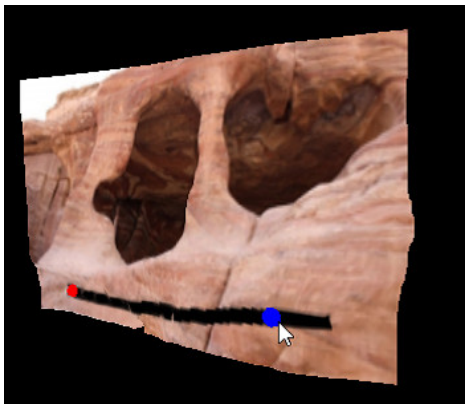
図 3.3: 式 3.5 を適用したデプスマップの比較



(a) 制御ハンドルの設置

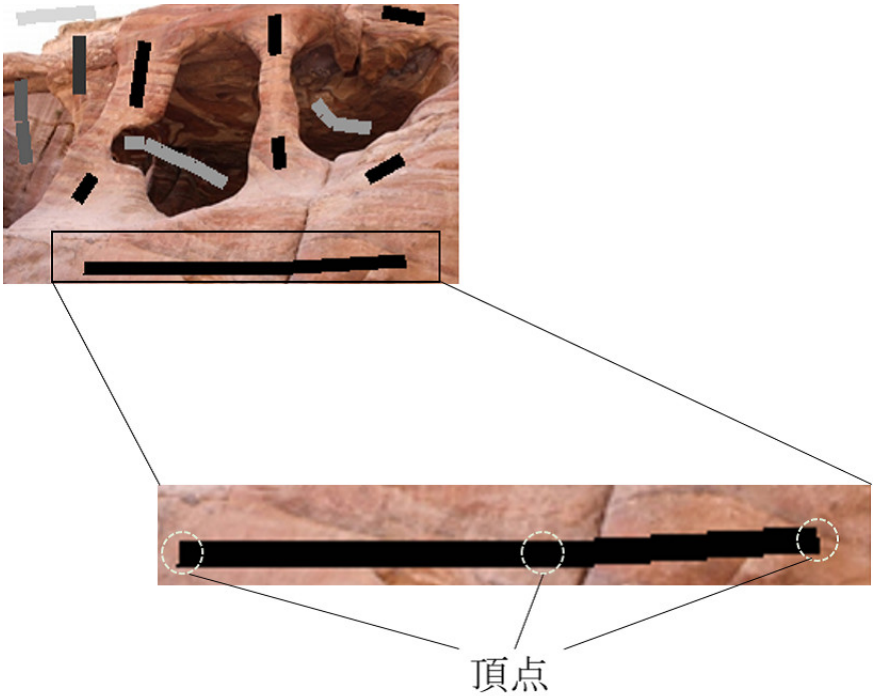


(b) 制御ハンドルによるデプス変更前

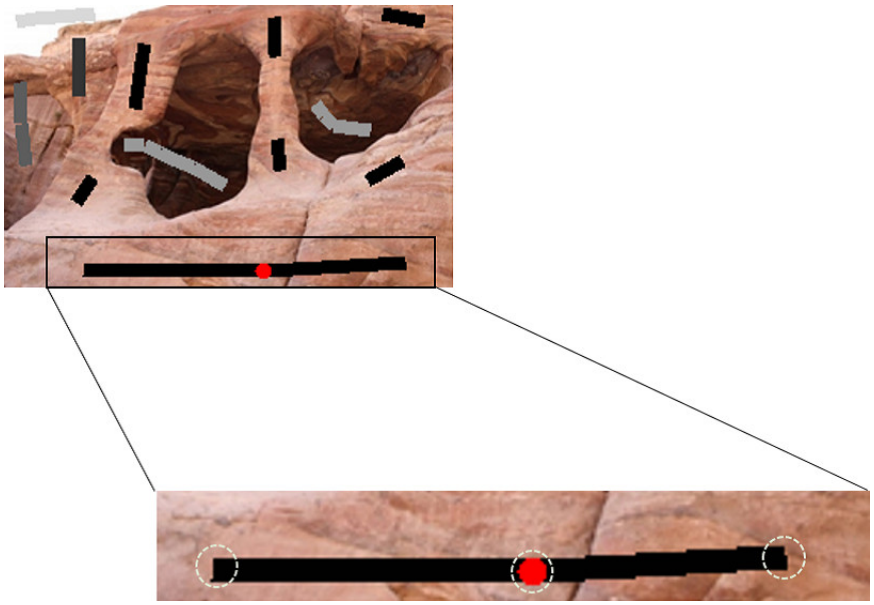


(c) 制御ハンドルによるデプス変更後

図 3.4: 制御ハンドルを用いたデプス変更の流れ。(b)、(c) では、デプス変更に使用している制御ハンドルを青色で示している。



(a) ストロークの頂点



(b) 制御ハンドルの配置

図 3.5: ストロークの構造イメージ図。各ストロークはそれぞれ頂点を保持しており、その頂点上に制御ハンドルを配置できる。

頂点上に、制御ハンドルを配置することができる (図 3.5(b) 赤丸部)。この制御ハンドルを動かすことで変形を行う。また、制御ハンドルによって奥行きを変化させる際、制約として奥行き方向への移動のみを許可している。これは奥行きを操作した際、モデルが歪まないようにするためである。

3.4.2 ストローク移動時の各頂点の動き

図 3.6、3.7 に、制御ハンドルで奥行きを操作した際の各頂点の移動イメージを示す。図 3.6(b)、3.7(b) 中の青丸で示される制御ハンドルを点線で示される矢印の方向に移動し、青の点線の丸で示される部分まで移動する。そうした場合、図 3.6(c)、3.7(c) のように、移動していない制御ハンドルの座標は固定され、(図 3.6(c)、3.7(c) 中の赤丸部分)。制御ハンドルが設置されていない頂点は、制御ハンドル間のデプスを補間するように移動する (図 3.6(c)、3.7(c) 中の緑丸部分)。

3.4.3 Sorkine らの手法 [8] によるストローク頂点の移動

ストロークの奥行きの変形には、Sorkine らの手法 [8] の二次元でのメッシュ変形の計算を利用している。図 3.8 に、Sorkine らの手法 [8] によるメッシュ変形の例について示す。図中の赤丸部分と黄丸部分が、本手法の制御ハンドルと対応する。我々の手法ではデプス値の変更のみを行うため、Sorkine らの手法 [8] での変形は奥行き方向のみに利用している。

Sorkine らの手法 [8] では、以下の誤差関数の $\mathbf{v}'_i$ について解くことにより、変形後の結果を求めている。

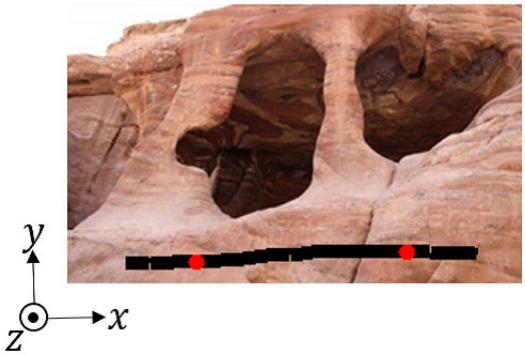
$$E = \sum_{i=1}^{n} \|T_i \mathcal{L}(\mathbf{v}'_i) - \mathcal{L}(\mathbf{v}'_i)\|^2 + \sum_{i=m}^{n} \|\mathbf{v}'_i - \mathbf{u}_i\|^2 \tag{3.6}$$

ここで、T_i は移動後の形状を保つための変換行列、$\mathcal{L}(\mathbf{v}_i)$ は点 $\mathbf{v}_i$ のラプラシアン座標、$\mathbf{v}_i$ は変換前の座標、$\mathbf{v}'_i$ は変換後の座標、$\mathbf{u}_i$ はユーザが指定した点の移動後の座標である。Sorkine らの手法 [8] では、$\mathcal{L}(\mathbf{v}_i)$ は以下のように定義されている。

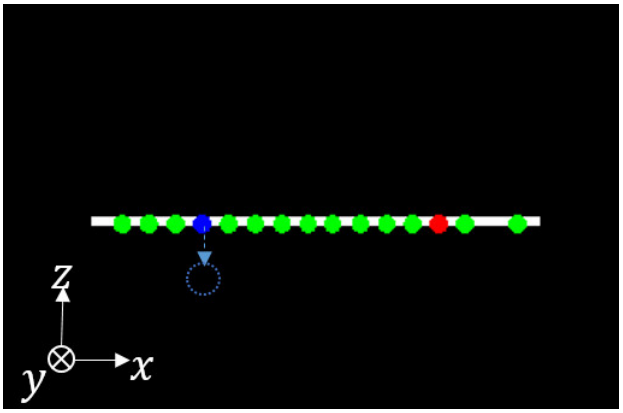
$$\mathcal{L}(\mathbf{v}_i) = \mathbf{v}_i - \frac{1}{d_i} \sum_{j \in \mathcal{N}_i} \mathbf{v}_j) \tag{3.7}$$

ここで、$\mathcal{N}_i$ は点 i の近傍の点の集合、d_i は $\mathcal{N}_i$ の数である。式 3.6 において T_i は未知の行列であり、式 3.6 を線型方程式で解くためには T_i が線形変換である必要があるが、一般に回転行列では線形変換で表せないため、以下のような行列で近似される。

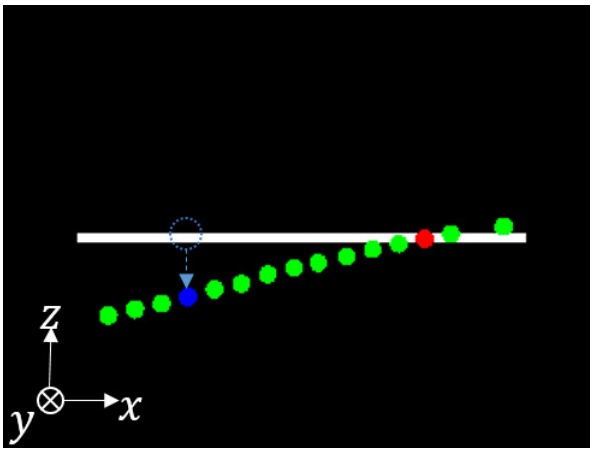
$$T_i = \begin{pmatrix} a & w & t_x \\ -w & a & t_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$



(a) 入力画像

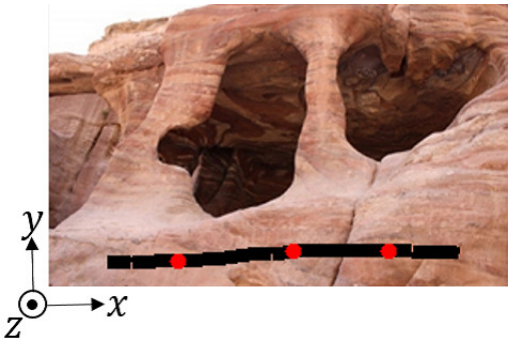


(b) 制御ハンドル移動前

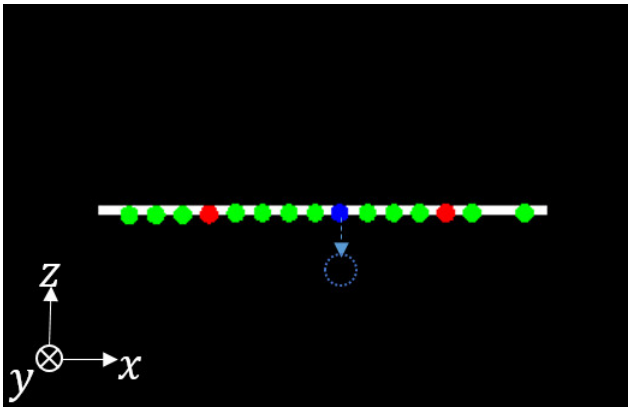


(c) 制御ハンドル移動後

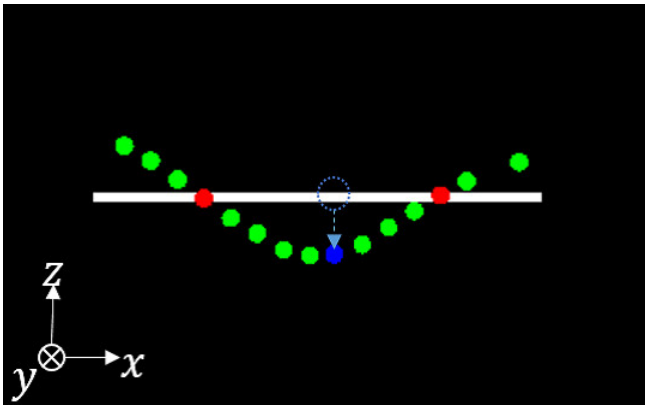
図 3.6: 二つの制御ハンドルによる各頂点の移動イメージ



(a) 入力画像



(b) 制御ハンドル移動前



(c) 制御ハンドル移動後

図 3.7: 三つの制御ハンドルによる各頂点の移動イメージ

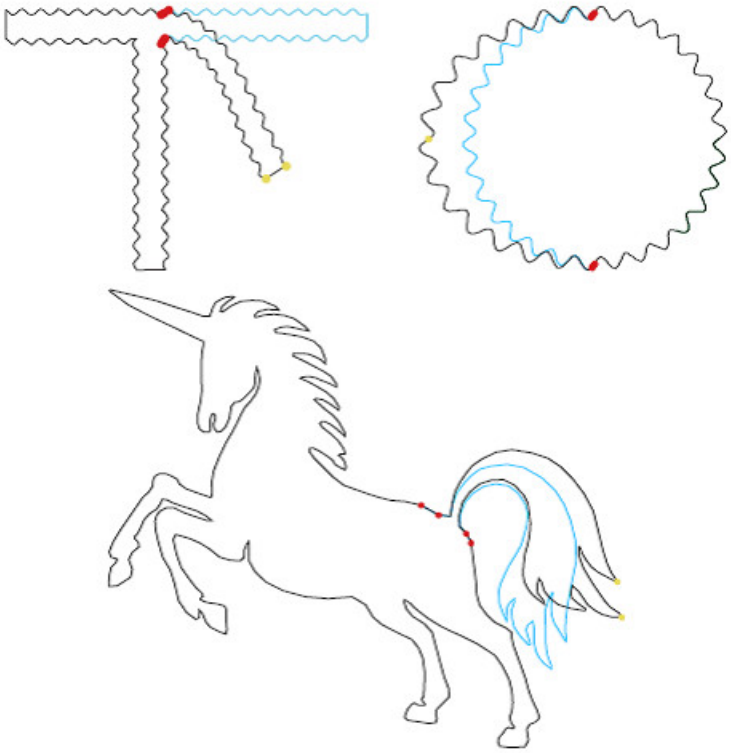


図 3.8: Sorkine らの手法 [8] によるメッシュ変形の例。青線で示されている部分が変形前の形状である。赤丸で示されている部分が固定されており、黄丸で示されている部分を動かして変形している。

出典: Laplacian Surface Editing[8]

この行列の未知変数 a、w、t_x t_y を求めることで T_i が決定される。変換行列 T_i は、以下の式によって求める。

$$\min_{T_i} \left(\left\| T_i \mathbf{v}_i - \mathbf{v}'_i \right\|^2 + \sum_{j \in \mathcal{N}_i} \left\| T_i \mathbf{v}_j - \mathbf{v}'_j \right\|^2 \right) \tag{3.9}$$

これによってストローク上の各頂点の奥行きを計算し、その結果を節のデプス伝搬の入力とすることでモデルを変形させる。

3.5 背景領域の生成

背景領域の生成には、Iizuka らの手法 [1] における不連続な箇所の推定と、遮蔽されている領域の推定を利用している。

不連続の箇所の推定には、Iizuka らの手法の [1] の以下の式を使用している。

$$w_d E_d + w_i E_i \tag{3.10}$$

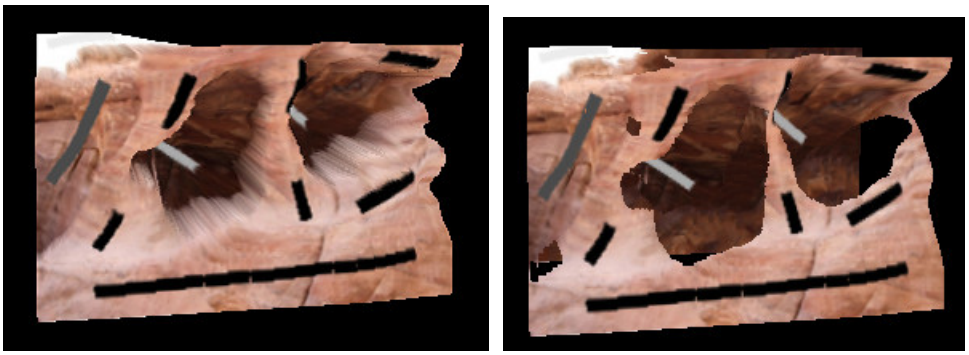
ここで、w_d、w_i はそれぞれデプスマップのエッジと入力画像のエッジの重みで、本手法では Iizuka らの手法 [1] と同様に $w_d = 0.8$、$w_i = 0.2$ という値を使用している。E_d、E_i はそれぞれデプスマップのエッジ、入力画像のエッジである。この式によって閾値処理を行うことで不連続となる箇所を検出する。

背景領域の生成は、Iizuka らの手法 [1] に基づいて以下の順序で計算される。

1. 不連続箇所を膨張させる。
2. デプス値からヒストグラムを生成し、閾値処理を行う。
3. 閾値に基づき、背景、前景、未知領域に分類した Trimap を生成する。
4. Trimap から、バイナリラベリング処理により遮蔽領域を抽出する。
5. 遮蔽領域を類似パッチ探索によって補間し、背景テクスチャを生成する。
6. 背景領域のデプスを、背景テクスチャと入力画像のデプスによって計算する。

これにより、背景領域が生成される。Trimap からのバイナリラベリングと類似パッチ探索は、Iizuka らの手法 [1] と同様にそれぞれ Bai らの手法 [9]、Barnes らの手法 [10] を用いている。背景領域を生成することで、自然な三次元シーンモデルを生成することができる。

図 3.9 に、不連続箇所に切り込みを入れ背景領域を生成する処理をしたモデルと、していない画像を示す。図 3.9(a) では洞窟の穴のような、不連続となるはずの箇所が連続的になってしまっているのに対し、図 3.9(b) では不連続となる箇所に切り込みが入り、背景が見えるようになっているため、実際の物体に近い三次元シーンモデルとなっていることがわかる。



(a) 背景領域処理をしたモデル画像　　(b) 背景領域処理をしていないモデル画像

図 3.9: 背景領域処理をしたモデルの比較

3.6　モデルの修正操作

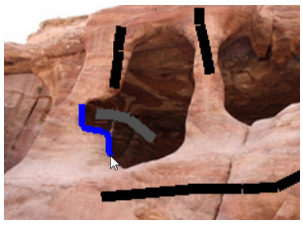
モデルの修正操作として、不連続箇所の手動での追加と、フリーハンド入力によるデプスの平滑化について示す。

3.6.1　手動での不連続箇所の追加

3.5 節において、モデルの不連続箇所に切り込みを入れる手法を示したが、図 3.10 のように、ユーザのデプス入力によっては上手くモデルに切り込みが入らない場合がある。そのため、不連続箇所に検出されなかった部分に対して手動で不連続箇所を指定できるようにした。

手動での不連続箇所の追加について、図 3.11 にイメージ図を示す。図での青色のストロークによって指定することで不連続箇所となり、モデルへの切り込みが生成される。指定は、入力画像、出力モデルどちらに対しても行えるようにしている。

図 3.10: 上手く不連続箇所として検出されなかった例



(a) 入力画像に対する不連続箇所の指定

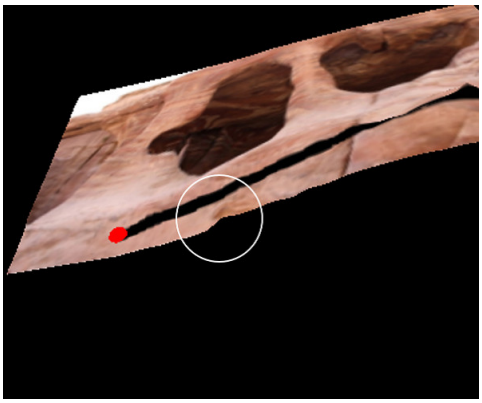
(b) 出力モデルに対する不連続箇所の指定

図 3.11: 不連続箇所の指定のイメージ図

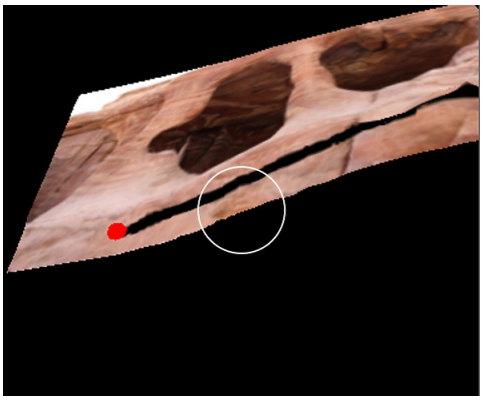
3.6.2 フリーハンド入力によるモデルのデプスの平滑化

デプス伝搬の計算を行った際に、画像によってはモデルの一部に凹凸ができる可能性がある。そのため、フリーハンドによるモデルのデプスの平滑化を実装した。実装されている平滑化は、単純な周囲のデプスの平均化によって行っている。モデルで平滑化したい部分をフリーハンドで指定することで平滑化を行うことができる。

図 3.12 に、実際にモデルに対して平滑化を行ったものについて示す。



(a) 平滑化前



(b) 平滑化後

図 3.12: モデルの平滑化の比較画像。白丸で囲っている部分が、平滑化され平らになっていることがわかる。

図 3.12(a) ではやや出っ張っている部分があるのに対し、平滑化後の図 3.12(b) ではその部分が改善されていることがわかる。

第4章　結果

本章では、実装したシステムと、提案手法によって生成されたモデル、既存手法との比較について述べる。

4.1　実装したシステム

4.1.1　実装言語と作成したユーザインタフェース

提案システムは、ライブラリとしてモデルの表示に OpenGL、行列計算に Eigen、画像の顕著性の計算に OpenCV を使用し、C++/CLI 言語により実装を行った。また、C++/CLI の PictureBox に OpenGL の表示を行うために、文献 [11] を参考にした。

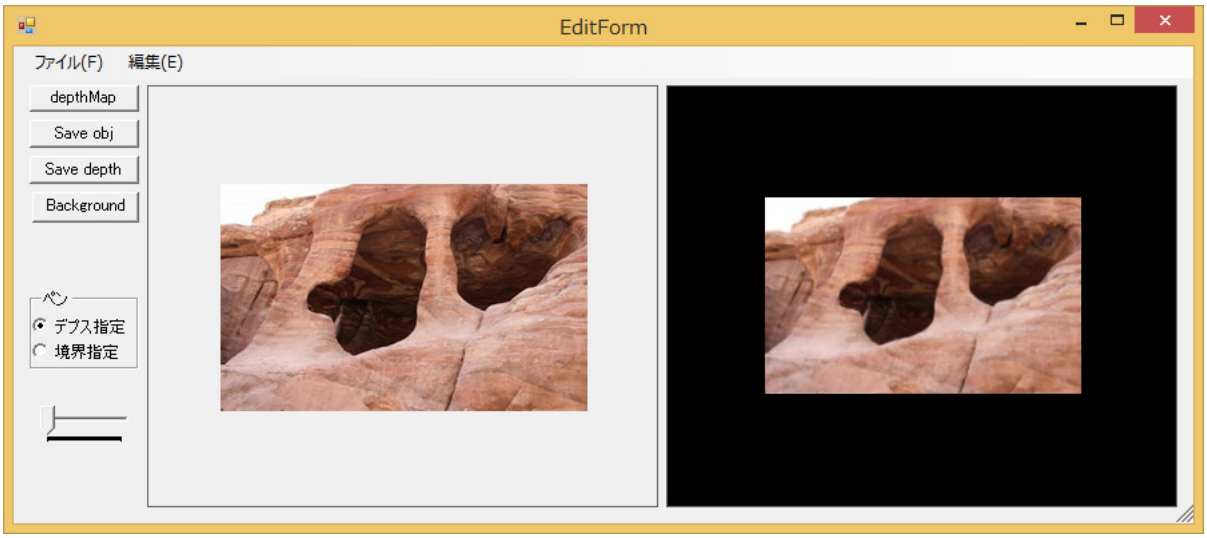


図 4.1: 実装したシステムのユーザインタフェース

図 4.1 に、本研究で作成したシステムを示す。左画面に入力画像、右画面に出力されるモデルが表示される。ブラシストロークによるデプス入力、制御ハンドルの設置はどちらの画面でも行える。デプス入力と不連続箇所の入力の切り替えは、ウインドウ左のラジオボタンによって切り替えることができる。

4.1.2 計算時間の計測

計算時間を、表 4.1 のスペックのコンピュータで実行し計測した。

表 4.1: 計算時間の計測環境

OS	Windows 7 Professional 64bit
CPU	Intel(R) Core(TM) i7-4930K CPU 3.4GHz
メモリ容量	32GB
GPU	NVIDIA GeForce GTX TITAN 4G

入力に使用した画像は 400×248 のサイズの画像を使用している。計測項目は、画像入力時の初期計算 (各ピクセルの重み係数の計算や入力画像の顕著性の計算など)、デプスの伝搬の 2 つである。表 4.2 に、計測結果を示す。

表 4.2: 計測結果

項目	平均時間 (秒)
画像入力時の初期計算	1.504
デプスの伝搬	0.2831

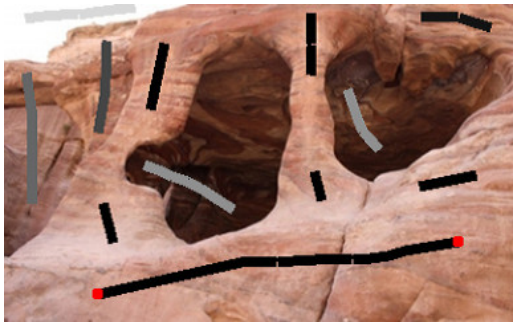
画像入力時の初期計算では約 1.5 秒とやや時間はかかるが、システムの起動に問題ない時間といえるだろう。デプスの伝搬の平均計算時間は約 0.3 秒と、リアルタイムで入力をモデルに反映させるには問題ない時間といえる。

4.2 既存手法との比較

図 4.2 に、本手法と Iizuka らの手法 [1] によって生成したモデルの比較を示す。使用した画像は 400×248 のサイズであり、デプス入力にしたストロークは、どちらも図 4.2(a) の入力を使用している。また、提案手法の入力においては、図 4.2(a) の赤丸部分で示される部分に制御ハンドルを配置し、右の制御ハンドルを画像に対して手前側に動かす操作を行っている。モデルは、MeshLab[12] で背景を白色に設定し、ライトをオフにして表示している。

図 4.2(d) と図 4.2(e) を比較すると、図 4.2(d) はモデルの下端が元の画像方向とほぼ平行で不自然に見えるのに対し、図 4.2(e) はモデルの下端は、制御ハンドルによる奥行きの操作を行ったことにより、下端に角度がついたことで図 4.2(d) に比べ自然な見た目となっていることがわかる。以上のことから、従来と同じストローク入力で、角度のついたモデルを生成できることがわかった。

図 4.3 に、本手法によるモデルと Iizuka らの手法 [1] によって、斜めの面を持つようにモデルを製作したものとの比較を示す。入力画像、図 4.3(b) での制御ハンドルの移動は図 4.2 で比較したときの入力と同じである。下端の形状は類似しているが、図 4.3(a) と図 4.3(b) を比較すると、図 4.3(a) のほうが下部分のストローク入力数が多くなっていることがわかる。



(a) 入力デプス

(b) Iizuka らの手法 [1] によるモデル

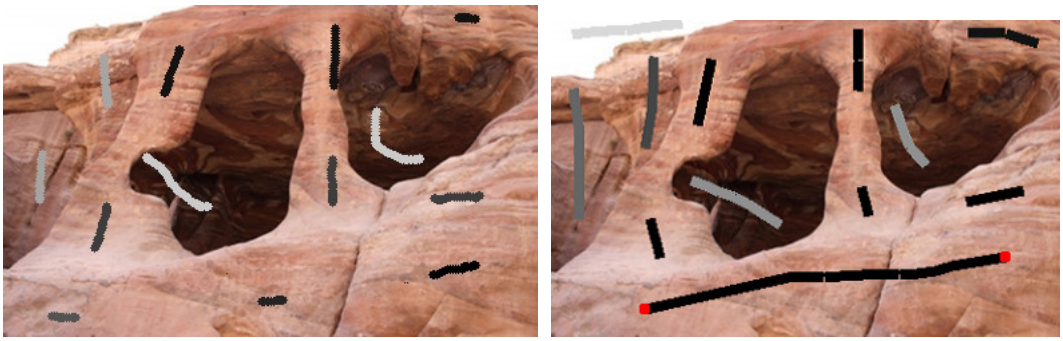
(c) 提案手法によるモデル

(d) 下から見た (b) のモデル

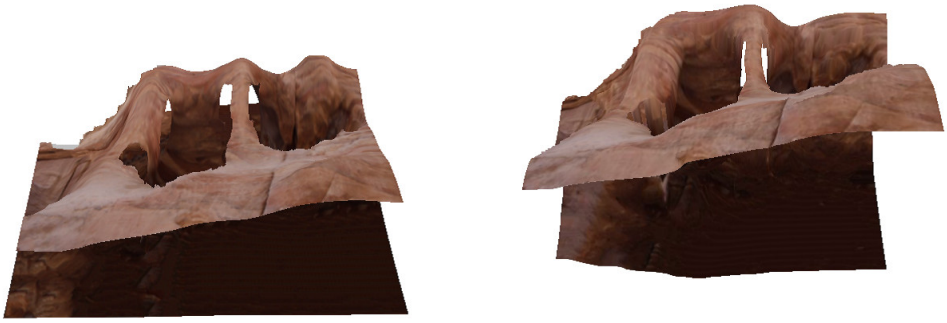
(e) 下から見た (c) のモデル

図 4.2: Iizuka らの手法 [1] との比較

以上のことから、少ないストローク入力によって従来の手法よりも詳細なモデルを生成できることがわかった。



(a) Iizuka らの手法でのデプス入力　(b) 提案手法によるデプス入力



(c) 下から見た (a) のモデル　(d) 下から見た (b) のモデル

図 4.3: Iizuka らの手法 [1] を用いて製作した斜めの面を持つモデルとの比較

4.3　生成したモデル群

図 4.4 に、本システムによって作成したモデルを示す。左がユーザ入力、右が生成されたモデルとなっている。ユーザ入力では、グレースケールで引かれている線がデプス入力に用いたストローク、青色で引かれている線が不連続箇所の指定に用いたストロークである。ストローク上の赤点はデプス操作に用いた制御ハンドルである。

図 4.4: 本手法によって作成したモデル
各入力画像の出典: 写真素材ぱくたそ、https://www.pakutaso.com

第5章　まとめと今後の展望

本章では、本研究の結論と、本研究の今後の展望、課題について述べる。

5.1　まとめ

本研究では、ブラシストロークでのデプス指定によるデプスマップの生成に加え、入力したブラシストローク上に制御ハンドルを配置し奥行き方向に制御ハンドルを操作することで、一つのストローク上でモデルの奥行きに変化を持たせる手法を提案した。これにより、ストロークのデプス入力のみでは作成が難しい角度のついた三次元シーンモデルを生成できるようになった。また、既存手法との比較を行い、既存手法よりも少ないストローク入力で詳細な三次元シーンモデルを生成することができるようになった。

5.2　今後の展望

現在の手法では、画像サイズが大きくなるとデプスマップの最適化計算に時間がかかり、インタラクティブな操作がしづらくなる問題がある。改善方法として、デプスの最適化計算を行う際に領域分割を行い、奥行きを変化させた領域付近のデプスマップのみを再計算する方法が挙げられる。

また、現在のシステムではユーザの使いやすさといった点について考慮がされていない。多くのユーザにシステムを利用してもらうにはシステムの使いやすさが重要であるため、ユーザビリティを考慮したシステムへの改良を行いたいと考えている。さらに、改良したシステムに対してユーザテストを行い、ユーザから得たフィードバックからインタフェースの改良を進めたいと考えている。

三次元シーンモデルの他にも、アナグリフのような三次元映像の生成にも本手法を適用できると考えられる。静止画だけでなく、動画へも本手法を適用することができれば、三次元映画といった三次元動画コンテンツへ応用することができる。これにより、製作に手間がかかる三次元動画の製作の手間を短縮するようになると考えられる。

謝辞

本研究を行うにあたり、ご指導、ご助言をいただきました本学大学院システム情報系の金森由博助教、三谷純教授に深く感謝いたします。また、プログラムの提供をしていただき、さらに研究の方針や手法についてのご助言をいただきました早稲田大学大学院基幹理工学研究科の飯塚里志助教、本研究への協力や論文の執筆にあたりご助言をいただいた非数値処理アルゴリズム研究室の皆様に深く感謝を申し上げます。

参考文献

[1] Satoshi Iizuka, Yuki Endo, Yoshihiro Kanamori, Jun Mitani, and Yukio Fukui. ”Efficient Depth Propagation for Constructing a Layered Depth Image from a Single Image”. Computer Graphics Forum (Proc. of Pacific Graphics 2014), 33, 7, 279-288, 2014.

[2] ストリートビュー - Google マップ, https://www.google.co.jp/intl/ja/maps/streetview/ (参照 2016-1-11)

[3] Youichi Horry, Ken-Ichi Anjyo, Kiyoshi Arai. ”Tour Into the Picture: Using a spidery mesh user interface to make animation from a single image”, Proc. SIGGRAPH 97, 225-232, 1997.

[4] Satoshi Iizuka, Yoshihiro Kanamori, Jun Mitani, Yukio Fukui. ”Efficiently Modeling 3D Scenes from a Single Image”, IEEE Computer Graphics & Applications. 2012.

[5] D. Hoiem, A.A. Efros, and M. Hebert, ”Automatic Photo Pop-up”, ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2005), 24, 3, 577-584, 2005.

[6] Wang, O. and Lang, M. and Frei, M. and Hornung, A. and Smolic, A. and Gross, M. ”Stereo-Brush: interactive 2D to 3D conversion using discontinuous warps”, Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling, 47-54, 2011.

[7] Anat Levin, Dani Lischinski, and Yair Weiss. ”Colorization using optimization”, ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2004), 23, 3, 689-694, 2004.

[8] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rossl, and H.-P. Seidel. ”Laplacian surface editing”, Eurographics Symposium on Geometry Processing, 175-184, 2004.

[9] Bai, X., Sapiro, G. ”Geodesic matting: A framework for fast interactive image and video segmentation and matting”. Int. J. Comput. Vision 82(2), 113-132, 2009

[10] Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B. ”PatchMatch: A randomized correspondence algorithm for structural image editing”, ACM Trans. Graph. (Proc. of SIGGRAPH) 28(3), 24:1-24:11, 2009

[11] .NET_Framework__OpenGL, http://tokyo-ct.net/usr/kosaka/for_students/DNetOpenGL/DNetOpenGL.html (参照 2016-1-8)

[12] MeshLab, http://meshlab.sourceforge.net/ (参照 2016-1-10)