平成 29 年度

筑波大学情報学群情報科学類

卒業研究論文

題目 エッジ強調に基づくイラスト画像のための対話的領域分割

主専攻　　知能情報メディア主専攻

著者　　谷島 拓実

指導教員 金森由博 遠藤結城 三谷純

要　旨

近年，細かいパーツに分割されたイラストを用いてアニメーションを作るツールが用いられているが，イラストをパーツに分割する作業は容易ではない．そこで本研究は，簡易な入力によってイラストを領域分割できるようにすることを目的とする．対話的な処理で画像を複数の領域に分割する既存手法を用いてイラストの領域分割を行うと，イラストの輪郭線がエッジとして認識されなくなること，領域の境界の位置がユーザの意図した位置にならないこと，領域分割の結果からパーツに分割してもアニメーションに用いることができないことといった問題がある．そこで本研究ではこの手法に対して，隣接領域との境界にエッジが現れやすくなるようフィルタを変更した．また，ユーザに境界線を入力させることで正しい境界線を指定できるようにした．さらに，パーツに分割する際にアニメーションに用いることができるように分割の仕方を工夫した．提案手法によって，より効率的に，ユーザの意図をより反映した結果が得られるようになった．

目 次

図 目 次

第1章　序論

1.1　研究背景

通常，キャラクターの二次元アニメーションを作るには，動作の開始から終了までの絵を複数枚用意する必要がある．しかし，最近のアニメーションの現場では，一枚のイラストを基にアニメーションを作成できる，Live2D や spine などの商用ツールが利用されている．これらのツールを用いると，細かいパーツに分割されたイラストを入力として，それぞれのパーツを連続的に変形させることでアニメーションを作ることができる．このようなツールは，多様で滑らかなアニメーション表現ができるが，入力するパーツは最初からパーツごとに描き分けるか，画像をパーツごとに分割する必要があり，手間がかかるといった問題がある．

1.2　研究の目的

本研究では 1.1 節で示したイラストのパーツ分割を容易にすることを目的とする．対話的な処理で画像を複数の領域に分割する手法に LazyBrush [15] がある．これは，グレースケール画像を入力として，ユーザが色のついたブラシツールで線を引くと，画像のエッジが領域の境界線として認識され，画像が領域ごとに色分けされる，という手法である．類似した手法と比較して，以下の優位性がある．

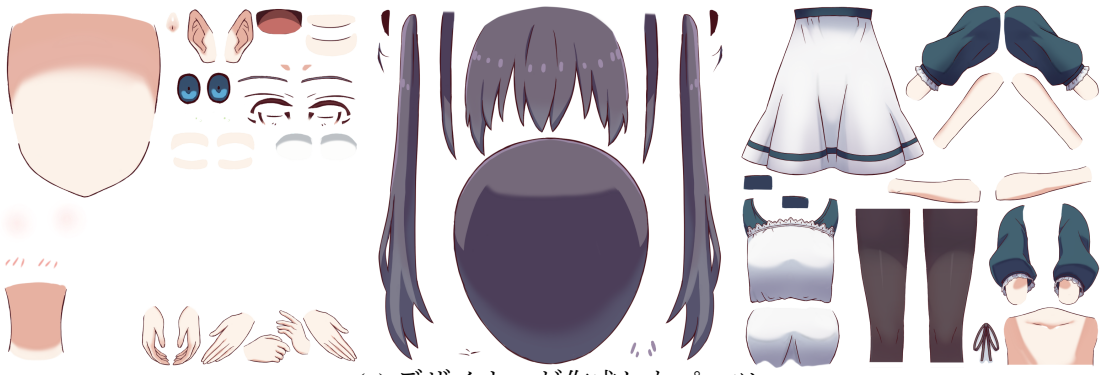
- エッジをはみ出した入力線を許容
- エッジの隙間からの色漏れを防ぐ
- 少ないユーザ入力で領域全体に色が伝播
- 入力画像のノイズを軽減

ただし，この手法はグレースケール画像を対象にしており，カラー画像に適用する際にはカラー画像をグレースケール画像に変換する必要がある．変換の際，元の色によっては輪郭線の色が薄くなり，領域分割に必要なエッジが検出されなくなる場合がある．また，アニメーション用途では関節などでパーツを分割する必要があるが，関節位置に必ずしも画像のエッジが存在するわけではないので，この手法では適切なパーツ分割は容易ではない．さらに，アニメーションの際に可動パーツの間に隙間が生じるのを防ぐため，可動パーツ同士には互いに重なり (以下では糊代と表記する) を持たせて分割しなければならない．そこで本研究では，この手法に対して以下の改良を加える．

- エッジを検出しやすいようフィルタ処理を改良

- ユーザによる明示的な境界線の指定
- 分割されたパーツの関節部分に糊代を追加

以上の改良により，イラストの領域分割を既存手法より容易に行えるようにする．



(a) デザイナーが作成したパーツ．

(b)Live2D によるアニメーション．

図 1.1: 部位ごとに細かく分割されたイラストとそれらを利用したアニメーション．
(画像の出典: `http://www.live2d.com/ja/products/cubism_editor`)

第2章　関連研究

2.1　前景抽出

画像の領域分割はコンピュータヴィジョンにおけるタスクの一つである．画像の前景抽出もその一種であり，さまざまな手法が提案されてきた．大きくハードセグメンテーション [9][3] とソフトセグメンテーション [4][12] の二つに分けられる．ハードセグメンテーションではピクセルに前景または背景のラベルを割り当てることで分類する．一方で，ソフトセグメンテーションでは，ピクセルに前景らしさまたは背景らしさの尤度を与え，その尤度で前景か背景かを決定する．前景抽出の手法は二値分類であり，本研究ではイラストのキャラクターを複数のパーツに分割するための多値の領域分割を目的としているため，これらの手法を用いることはできない．

2.2　画像の色付け

近年，グレースケール画像への対話的な色付けをする手法が提案されてきた．これらの手法は，領域の境界を求め複数のユーザ入力を拡げる手法であるため，多値の領域分割であるといえる．自然画像への色付け [8][5][16] は広く研究されてきたが，これらをイラストの色付けに用いると，テクスチャのない領域やコントラストの強いエッジによってうまくいかない．Sýkora ら [14] は白黒の漫画を対象にした色助手法を提案したが，エッジの隙間から色が領域の外に漏れることがある．Qu ら [11] の手法では領域の外に色が漏れる問題を解決したが，色線の伝播が領域内のエッジを越えられず途切れる．LazyBrush では画像のエッジをハード制約として扱わずにソフト制約として扱うことでこれらの問題を解決している．

本研究の目的はグレースケール画像の色付けのような単純な多値分類ではなく，イラストのキャラクターを身体のパーツによって意味的に分割することである．二次元画像のアニメーションのための領域分割のために LazyBrush の機能を改良する．

第3章　ベース手法

1.2 節で示したように，本研究では LazyBrush をベースに改良を行う．本節では，この手法の概要を以下に説明する．

この手法は多値の領域分割の手法である．ユーザは領域分割したい画像を入力し，その画像は内部処理でエッジ検出される．エッジ検出した画像に対してユーザはユーザ入力として色線を書き込む．その色線の色を領域に伝播させることでグレースケール画像の色付けを行う．手法の詳細について以下で説明する．

3.1　エネルギー関数

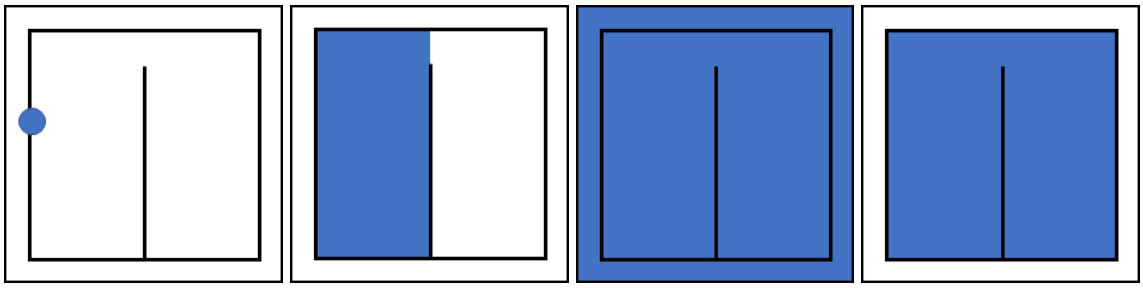
入力グレースケール画像 I，ピクセル集合 $\mathcal{P}$, 4 近傍 $\mathcal{N}$，ユーザ入力およびその色 $\mathcal{C}$ を定め，以下のエネルギー関数を最小化するピクセル p のカラー c_pを求める．

$$E(\mathcal{C}) = \sum_{\{p,q\}\in\mathcal{N}} V_{p,q}(c_p, c_q) + \sum_{p\in\mathcal{P}} D_p(c_p), \tag{3.1}$$

平滑化項 $V_{(p,q)}$はピクセル p とその近傍のピクセル q に割り当てられる色の連続性を示すエネルギーを表す．この項は領域の境界では小さくなり，連続する領域では大きくなるように設計する．データ項 D_pはカラー c_pをピクセル p に割り当てるエネルギーを表す．値を小さくすればユーザ入力すべてから色を拡げるハード制約となり，値を大きくすれば領域からはみ出したユーザ入力を無視するソフト制約となる．ここでは，ユーザ入力が正確でないことを許容できるようにするため，ソフト制約とする．図 3.1(a) のような入力をした場合，(b) 平滑化項が小さくなれば色が領域全体に拡がらない．また，(c) データ項が小さければ色線がはみ出ているために背景領域まで色が拡がる．(d) のような結果を得るために，各項を適切に定めなければならない．よって均質な領域では色が拡がり，そうでない場所で関数が最小となるよう各項を定義する．

3.2　平滑化項

この項はピクセル p, q 間の色の不連続性をなくすためのものである．輪郭線は黒に近い色で表されるため，色の伝播を切るのに最もふさわしい場所は元の画像のグレースケール値が小さいピクセルである．よって，エネルギー $V_{p,q}$を以下のように定める．



(a) 入力　(b) 平滑化項が小さい場合の結果　(c) データ項が小さい場合の結果　(d) 目標とする結果

図 3.1: 平滑化項とデータ項による影響.

$$V_{p,q}(c_p, c_q) \propto \begin{cases} I_p & (c_p \neq c_q) \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

I_pはピクセル p におけるグレースケール値である 色の伝播は 3.1 式が小さい時に切れるので，色の伝播が切れすぎないために，$c_p \neq c_q$の場合は $V_{p,q}$ がゼロにならないようにする必要がある．また，$c_p = c_q$ の場合，グレースケール値がゼロとなる輪郭線を持つ領域は 3.1 式の最小化に寄与しない．このような領域は色の伝播が途切れやすく，最終的な色付けの結果に穴を生じさせる可能性がある．対照的に，平滑化項がゼロでなければ穴のない領域になるが，白い領域を通って意図せず色が拡がる可能性もある．この欠点を解決するため，白い画素を通る境界線に高エネルギーを割り当てるようにする必要がある．この値の予想値は画像 I の周囲長である．これにより，w を画像の幅，h を画像の高さとし，$K = 2(w + h)$ として，グレースケール値の区間を $\langle 0, 1\rangle$ から $\langle 1, K\rangle$ に変更してマッピングする．二値画像に近ければマップは線形になるが，図 3.2(a) の箱側面の陰影のような柔らかいタッチがあると色の伝播が短く切れる問題が起こる．

このために，コントラスト向上の非線形マッピングや白黒画像のアウトライン検出の手法や，Colorization of black-and-white cartoons [13] での輪郭線検出に用いられる手法を適用できる．ここでは，Laplacian of Gaussian (LoG) フィルタを使用する．LoG フィルタでは画像強度の 2 次微分を推定し，そのゼロ交差はエッジ位置と曲率の高い場所への極大値となる．LoG フィルタによって入力画像を前処理する (LoG(I) と表す) ことで，負の応答が 0，正の応答が $\langle 0, 1\rangle$ になるように正の値 s でスケーリングされた値になり，新しい画像 $I_f = 1 - \max(0, s \cdot \mathrm{LoG}(I))$ を生成する．この処理によって，輪郭線のコントラストが強調され，均質領域の内部の強度は元のままで白色になる（図 3.2）．最終的に，I_fは $\langle 1, K\rangle$ で線形にマッピングされ，(2) 式の I_p に代わって平滑化項 $V_{p,q}$ に用いられる．

3.3 データ項

データ項 D_pは通常，パターンや強度の類似性などの画像ベースの値を用いて決められる．これは，色とパターンまたは色と強度の一対一対応を必要とする．しかし，手書きの画像ではパター

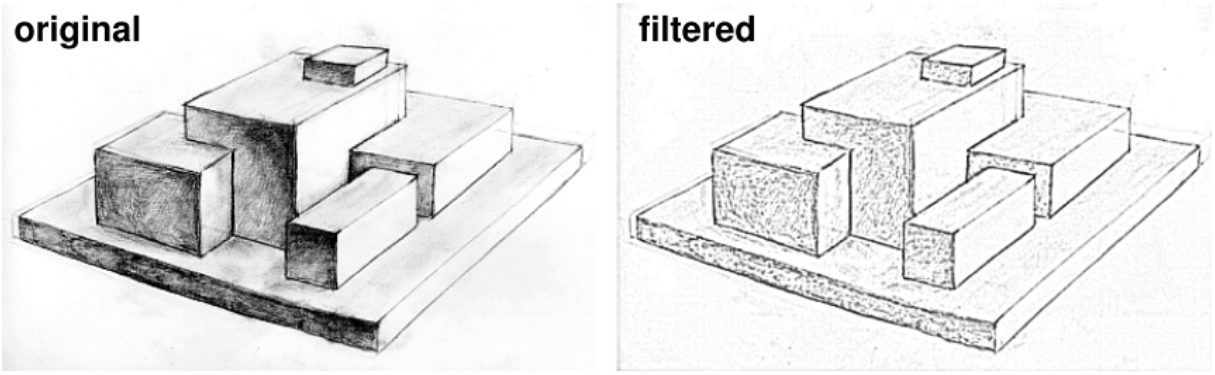


図 3.2: 入力画像 (左図) と LoG フィルタをかけノイズを軽減しエッジを強調した画像 (右図).
(画像の出典: 文献 [15])

ンと強度が反復されることは典型でなく，一対一対応はまれである．したがって，LazyBrush では画像ベースの値ではなくユーザ主導のデータ項を利用する．

以下の式を設定することでペナルティを追加し，ソフト制約にすることができる．

$$D_p(c_p) = \lambda \cdot K \tag{3.3}$$

$\lambda \in 0, 1$ はユーザが設定する定数である．K は，データ項と平滑化項の影響のバランスをとる白いピクセルの不連続性のエネルギーであり，3.2 項で定めたものと同じ値を使用する．また λ はユーザ入力とその強さを示し，$\lambda = 1$ ならばユーザ入力が無視され，$\lambda = 0$ ならばハード制約となる．ここでは，$\lambda = 0.95$ とした．

3.4 最小化

ここでは (3.1) 式の最小化について説明する．平滑化項 $V_{p,q}$ はピクセルのグレースケール値のみに依存し，塗られる色 (カラーラベル) には依存しない．したがって，このエネルギー関数の最小化は Potts モデル [10] を満たす．Boykov ら [1] が示しているように，(3.1) 式のようなエネルギー関数の最小化は，頂点の集合 $\mathcal{V} = \{\mathcal{P}, \mathcal{C}\}$ とエッジの集合 $\mathcal{E} = \{\varepsilon_p, \varepsilon_c\}$ からなる無向グラフ $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ 上の多方向カット問題を解くのに等しい (図 3.3).

頂点 $\mathcal{V}$ はピクセルの集合 $\mathcal{P}$ と色の集合 $\mathcal{C}$ からなる．それぞれのピクセル $p \in \mathcal{P}$ はエッジε_p を通して 4 近傍と接続され，このエッジ ε_pは $c_p \neq c_q$ の場合に平滑化項と等しい重み $w_{p,q} = V_{p,q}$ を持つ．補助エッジ ε_cは色の集合 $\mathcal{C}$ とピクセルをつなぐエッジであり，これは重み $w_{p,c} = K - D_p(c)$ を持つ．よってハード制約では $w_{p,c} = K$ であり，ソフト制約では $w_{p,c} = (1 - \lambda)K$ となる．LazyBrush におけるグラフは非常に疎なので，ε_c が少なくなり，多くのピクセルで $D_p = K$ となって $w_{p,c} = 0$ となり，対応する ε_c は余分になる．ユーザが定義した以外に色の集合への接続はないので、結果のラベルは常にシードに接続される．

カラーターミナルが 3 つ以上になると最大フロー/最小カット問題は NP 困難となる．そこで，

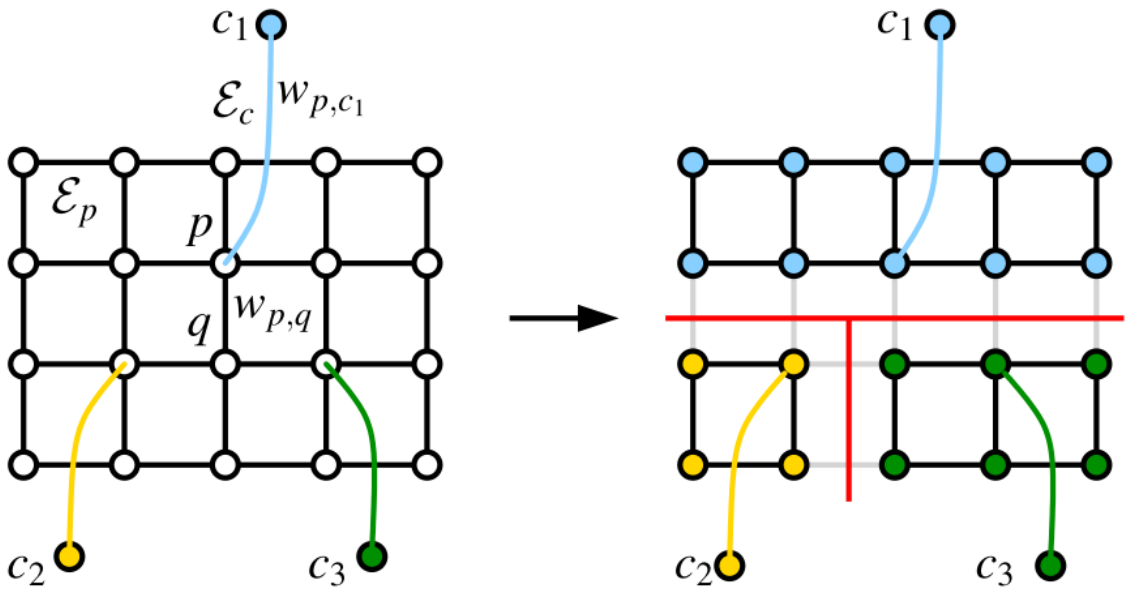


図 3.3: 左図の色が入力された場合の多方向カットによるラベルづけ (右図).
(画像の出典: 文献 [15])

Sýkora らは接続ラベルを保証するグラフトポロジーを利用した，貪欲多方向カットアルゴリズムを提案した．これは，N 個（N は色数）のすべての最大フロー最小カット問題よりも少ない計算量であり，単純な階層的方法である．最大フロー/最小カット問題のサイズを徐々に縮小し，単純な問題に落とし込むことによって大幅な速度の向上が得られる．具体的には，以下の手順を踏む (図 3.4).

1. 有効なカラーラベル $\mathcal{C}$ のセットとラベルのないピクセルのマスクの集合 $\mathcal{M}$ を初期化.
2. $\mathcal{M}$ の中で，1 つのカラーラベル c_rのみで色線と交差する，ラベルのない領域 $\mathcal{R}$ をすべて検索する. $\mathcal{M}$ の中のそれぞれの $r \in \mathcal{R}$ についてラベルを c_rに設定し, $\mathcal{M}$ の中にラベル c_rの色線を含む領域が他になければ, $\mathcal{C}$ から c_rを削除する.
3. $\mathcal{C}$ が空ならば終了.
4. 任意のカラーラベル $c \in \mathcal{C}$ を選択.
5. $\mathcal{M}$ の中のラベルのない全てのピクセルからグラフ $\mathcal{G}$ を作る.
6. カラーラベル c のシードとなるピクセルをターミナル $\mathcal{S}$ に接続し，カラー $\mathcal{C} - \{c\}$ のシードとなるピクセルをターミナル $\mathcal{T}$ に接続.
7. ソース $\mathcal{S}$ とシンク $\mathcal{T}$ で $\mathcal{G}$ の最大フロー最小カット問題を解く.
8. 対応するグラフの頂点がターミナル $\mathcal{S}$ に割り当てられたピクセルでは，マスク $\mathcal{M}$ 中のラベルを c に設定する.
9. カラーラベル c を $\mathcal{C}$ から除去し，手順 2 へ移行する.

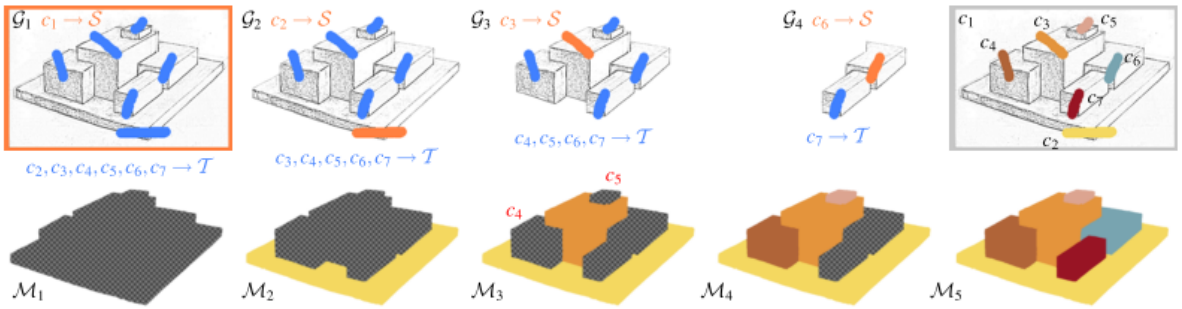


図 3.4: ループ処理による多方向カットアルゴリズムの様子.
(画像の出典: 文献 [15])

3.5 イラストの領域分割を行う際の問題点

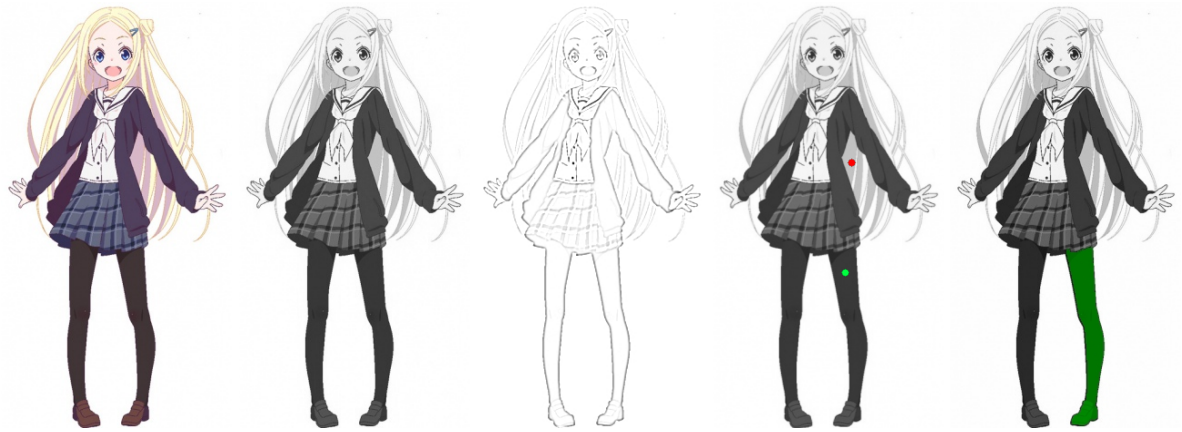
本研究の目的はカラーイラストのキャラクターを身体のパーツに応じて領域分割することである．LazyBrush をキャラクターの領域分割にそのまま用いると下記の問題が発生する．

LazyBrush はグレースケール画像を対象にしているため，カラー画像はグレースケール画像に変換する必要がある．この変換の際，明るい色の領域ではエッジが弱くなり，LazyBrush のエッジ検出に失敗する．すると，その領域では本来の隣接領域との境界がなくなってしまい，意図した色の伝播が行われなくなってしまう．その例を図 3.5 に示す．(a) 入力画像を (b) グレースケール画像に変換し，(c) LoG フィルタをかける．(c) では髪の毛と背景の境界線が薄くなり，エッジとして検出しづらくなっている．そのため，(d) のように入力をした場合，緑の点を入力した左足には色が拡がるが，赤の点を入力した髪では髪と背景領域と区別できず，(e) のように領域分割は失敗する．

アニメーション作成の際にはイラストの人物の腕や脚を動かすことがあり，腕や脚は関節部分のみで曲がるため，関節付近で領域を分割したい．しかし，LazyBrush による領域分割では，関節位置にエッジがなければ領域の境界線の位置が関節の位置からずれてしまう．LazyBrush による領域分割の結果を図 3.6 に示す．(c) の肩部分に注目すると，上腕と肩の境界線は肩から水平に伸びており，これは関節の位置とは言えない．

イラストからアニメーションを作るにはイラストを部位ごとにパーツとして分割する必要がある．LazyBrush でユーザが入力した色はエッジの手前までしか伝播せず，エッジの上に色は塗られていないため，色の塗られた領域のみを抽出すると輪郭線を抽出できない．色の塗られた領域のみを抽出した結果を図 3.7 に示す．(c) と比べると，(b) では顔の輪郭線部分が薄くなっていることがわかる．

また，イラストを分割してアニメーションを作る場合，可動パーツは糊代を含んで領域分割されなければならない．図 3.8 は Live2D で利用できるようパーツを分割した例である．上腕・前腕ともに肘部分 (赤枠) を糊代としているのがわかる．

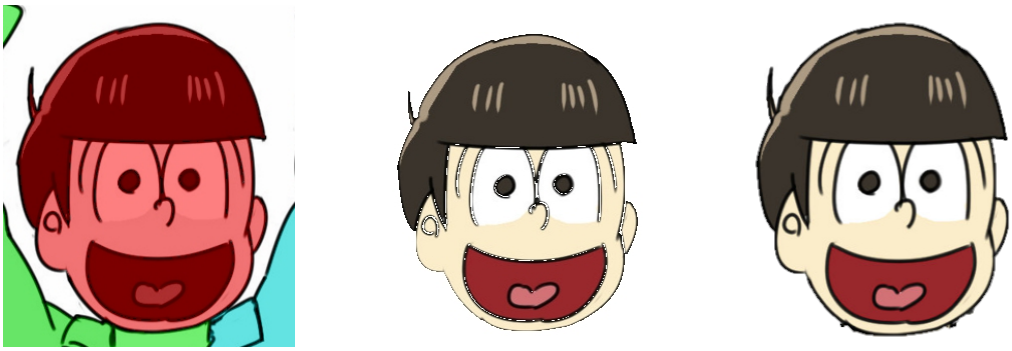


(a) 入力画像　(b) グレースケール画像への変換結果　(c)LoG フィルタリングの結果　(d) ユーザ入力　(e) 領域分割の結果

図 3.5: エッジ検出ができないことによる領域分割の失敗例.
(画像の出典: https://www.oricon.co.jp/news/2036333/photo/8/)

(a) 入力画像　(b) ユーザ入力　(c) 領域分割の結果

図 3.6: 関節位置にエッジがない場合の領域分割の結果.
(画像の出典: http://chemi-mizuki.hatenablog.com/entry/2015/12/03/120000)



(a) 領域分割の結果　(b) 赤に塗られた領域を抽出した結果　(c) 正解画像

図 3.7: 色の塗られた領域を抽出した場合の結果.
(画像の出典: http://chemi-mizuki.hatenablog.com/entry/2015/12/03/120000)

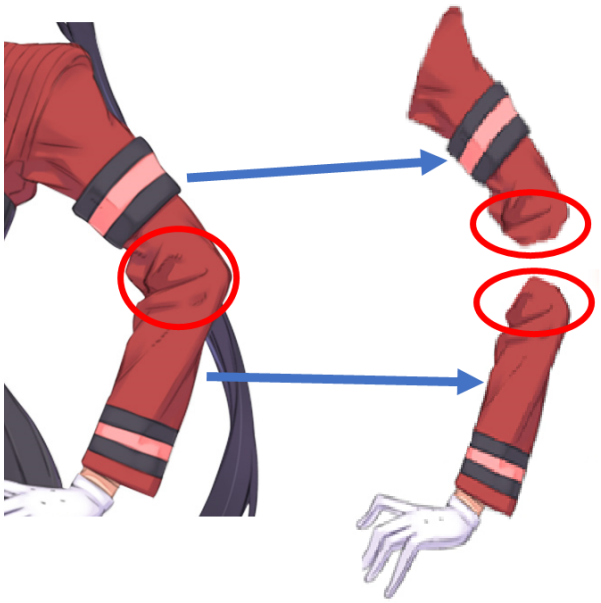


図 3.8: Live2D で利用する分割されたパーツ.
(画像の出典: https://cgworld.jp/interview/1306-live2d-summonnight5.html)

第4章　提案手法

本章ではLazyBrushをベースとして，第3.5節で述べた問題点を解決するために提案する手法について述べる．

4.1　フィルタの変更

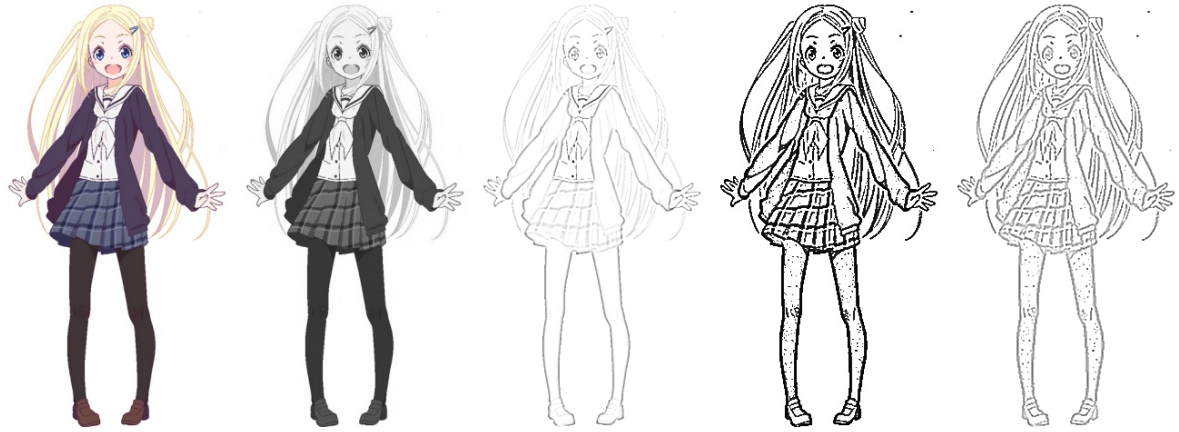
輪郭線が薄くなる問題を解決するために，フィルタ処理に2種類の変更を加えた．以下でそれぞれの手法について説明する．

4.1.1　ヒストグラム平坦化

エッジ強調の手法の一つにヒストグラム平坦化がある．グレースケール画像の濃度値をヒストグラムで表し，度数を平坦に近づけることで画像に抑揚をつける手法である．この手法をLoGフィルタリング後の画像に適用することによって，薄くなった境界線を強調してエッジとして認識できるようにする．その結果を図4.1に示す．(a)は入力画像，(b)は(a)をグレースケール画像に変換した結果であり，(c)は(b)にLoGフィルタをかけた結果，(d)は(c)にヒストグラム平坦化を用いた結果，(e)は(d)にLoGフィルタをかけた結果である．(c)では髪の毛の輪郭線が薄くなっているが，(d)では輪郭線が濃くなっている．ただし，LoGフィルタで軽減したノイズが(d)で強調されてしまっているため，このままではユーザの意図しない部分がエッジとして認識される恐れがある．そこで，(e)のように再びLoGフィルタをかけることでエッジを強調しつつノイズを軽減することができる．

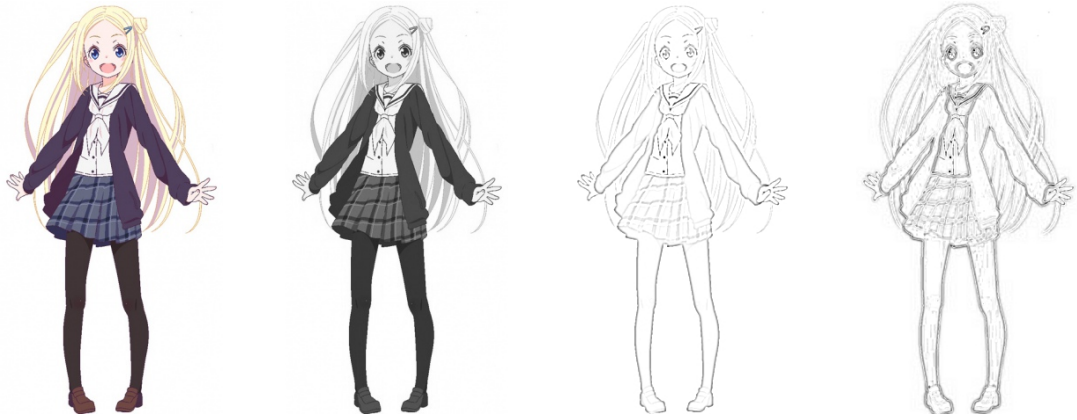
4.1.2　彩度の利用

グレースケール画像に変換する際，これまではカラー画像の明度を用いてグレースケール画像に変換していた．ここで，HSV色空間のH(色相)，S(彩度)，V(明度)のうちの彩度と明度を利用することによってフィルタリングによる問題を軽減できるかを試した．具体的には，RGB画像を読み込んでからHSV色空間に変換し，彩度画像と明度画像に対してLoGフィルタをかけその結果を乗算したものを，明度のみにフィルタをかけたものと比較した．その結果を図4.2に示す．(c)と比べると，(d)では全体的に輪郭線が強調されている．しかし，顔の部分に注目すると，口元や目元で輪郭線の周りにノイズが発生してしまっており，それらのノイズによって領域が切れる可能性がある．



(a) 入力画像　(b) グレースケール画像への変換　(c) LoG フィルタリングの結果　(d) (c) + ヒストグラム平坦化　(e) (d) + LoG フィルタリング

図 4.1: フィルタリングに変更を加えた場合の結果の違い．
(画像の出典: https://www.oricon.co.jp/news/2036333/photo/8/)



(a) 入力画像　(b) グレースケール画像への変換　(c) 明度を用いた場合の結果　(d) 明度と彩度を用いた場合の結果

図 4.2: 彩度を用いることによるフィルタリング結果の違い．
(画像の出典: https://www.oricon.co.jp/news/2036333/photo/8/)

4.2　境界線入力機能

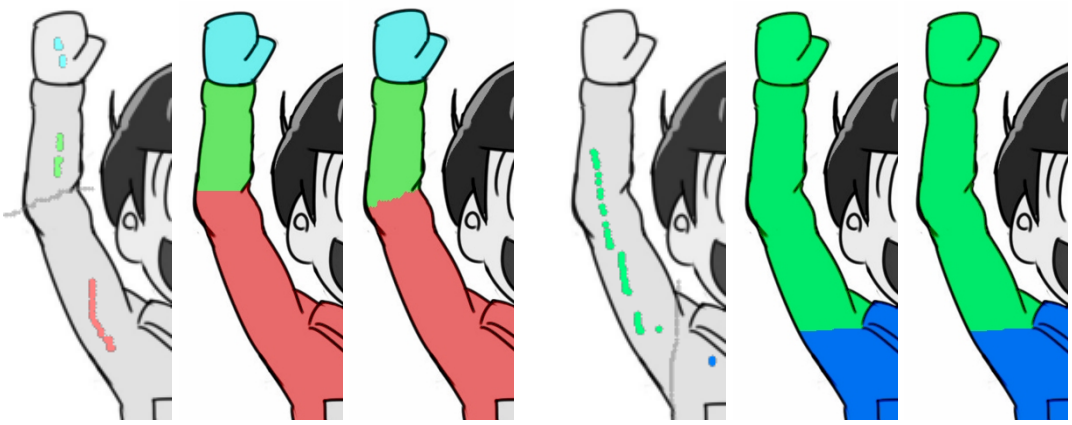
領域分割を行う際に，ユーザに線を入力させることで，その線が領域の境界線になるように補正を加えた．実装した 2 つの手法について説明する．

4.2.1 領域の伝播による入力結果の補正

領域分割後に，ユーザが境界線となる線を入力するとその線まで色が拡がるように補正を加えた．具体的には，以下の手順で実行される．

1. 入力した境界線と最も多くピクセルが重なっている領域を自動的に検出．
2. 境界線で分けられた領域の大小をピクセル数で比較．
3. 小さい方の領域に向かって境界線の各ピクセルから境界線との垂線を伸ばす．
4. 垂線の先の異なるラベルの領域を多数決で選択し，その領域を境界線まで拡げる．

この機能の実行例を図 4.3 に示す．それぞれ左の画像から，ユーザ入力，提案手法を実行する前の結果，提案手法を実行した後の結果であり，ユーザ入力の灰色の線が境界線の入力線である．(a) では境界線まで領域を拡げることができているが，(b) では結果が変化しない．これは，境界線の入力線が青の領域のみと交差していると判断され，各ピクセルから垂線をのばしても緑の領域が発見されず起こったものであると考えられる．この問題を解決するためには，拡げる領域をユーザが指定すればよいが，ユーザの手間は増えてしまう．

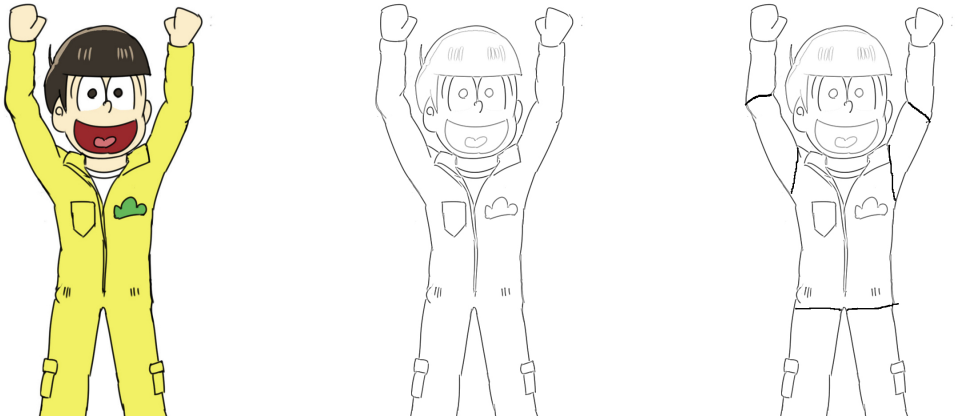


(a) 領域の伝播の成功例　　(b) 領域の伝播の失敗例

図 4.3: 境界線入力による領域の伝播の実行例．

4.2.2 フィルタ画像の編集による境界線入力

フィルタ処理をした画像にユーザが境界線を加えられるようにした．I_f が更新されるため，エネルギー関数の平滑化項が再計算される．この機能を用いた結果を図 4.4 に示す．(b) では関節の位置にエッジが存在しなかったが，(c) では境界線を入力することで関節の位置にエッジを足すことができている．



(a) 入力画像　(b)LoG フィルタリングの結果　(c) (b) に境界線を入力した結果

図 4.4: 境界線入力によってフィルタ画像に加えられたエッジ.
(画像の出典: http://chemi-mizuki.hatenablog.com/entry/2015/12/03/120000)

4.3 パーツ分割

領域分割の結果から，入力画像をアニメーションに使えるようをパーツに分割する．また，パーツに分けたイラストに含まれるジャギーを軽減する処理を加える．

4.3.1 エッジを含むパーツ分割

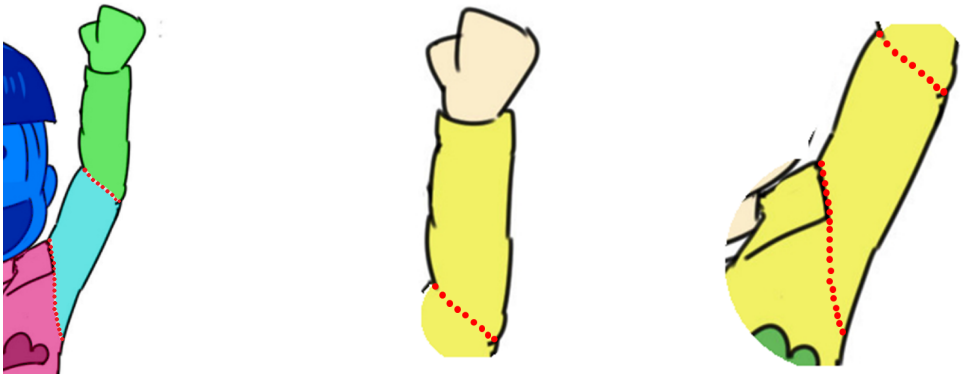
色の塗られていないエッジ部分もパーツに含めるための手法を提案する．本研究では，5×5 サイズのカーネルを用いたモルフォロジー演算によって色の塗られた領域を膨張させ，色の塗られていない領域をパーツに含めるようにした．この処理は領域分割後に自動で行われ，色ごとにパーツに分けて保存される．提案手法によるパーツ分割の結果を図 4.5 に示す．輪郭線が薄くならずに顔のパーツを抽出できるが，顔に含まれない襟部分の輪郭線も抽出されている．膨張を小さくすることで本来パーツに含まれない輪郭線を抽出しづらくなるが，輪郭線が太いイラストの場合には輪郭線をすべて抽出できずに薄くなってしまう．したがって，この方法ではイラストによってカーネルサイズを変える必要がある．

4.3.2 糊代を付加したパーツ分割

本研究では，ユーザが境界線を引いて分割した隣接領域間で，システムが自動的に可動部分に糊代を付加する機能を実装した．ここでは，糊代は境界線を囲む円内の領域とする．その結果を図 4.6 に示す．ユーザ入力に対する領域分割の結果が (a) であり，提案手法によって分割された前腕のパーツが (b)，上腕のパーツが (c) である．赤の破線はユーザが指定した境界線である．肘周

(a) 領域分割の結果　(b) 提案手法によって領域を抽出した結果　(c) 正解画像

図 4.5: 色の塗られた領域を膨張させてパーツを抽出した場合の結果.
(画像の出典: http://chemi-mizuki.hatenablog.com/entry/2015/12/03/120000)



(a) 領域分割の結果　(b) 緑の領域に糊代を付加した結果　(c) 青の領域に糊代を付加した結果

図 4.6: 可動部分に糊代を追加したパーツ分割.

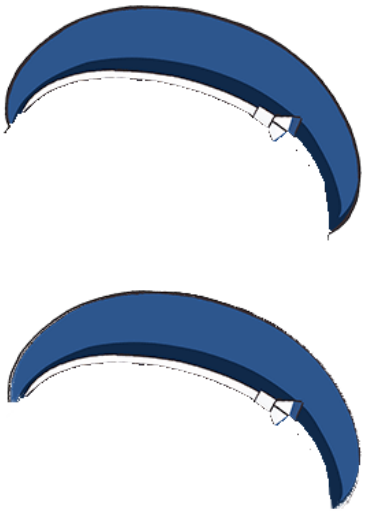
辺は互いに一部のみを切り取れているが，(c) の肩部分は襟や首の一部まで抽出してしまっている．可動部分を境界線を囲む楕円とすれば抽出する隣接領域が大きくなりすぎるのを防ぐことができると思われる．

4.3.3　アルファマッティングによるジャギー処理

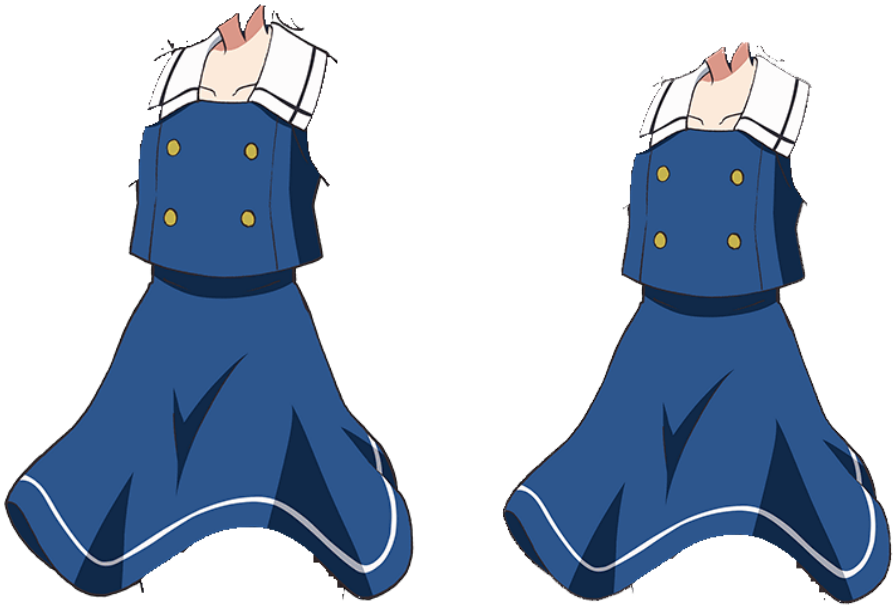
パーツに分割した際に生じるジャギーを処理するためにアルファマッティング [6] を行った．アルファマッティングは前景部分を抽出する手法であり，この手法を用いることでパーツに含まる部分のみを抽出できる．提案手法によってパーツに分けたイラストにアルファマッティングを適用

した結果を図 4.7 に示す．(a) は元画像，(b) の上側と (c) の左側の画像は提案手法によって切り分けたパーツであり，(b) の下側と (c) の右側の画像はそれぞれのパーツにアルファマッティングを行った結果である．抽出したパーツとアルファマッティングを適用した結果を比べると，アルファマッティングの結果ではパーツに含まれない輪郭線が軽減されていることがわかる．しかしながら (b) 下側の画像を見るとパーツに含まれるべき輪郭線も細くなってしまっていることがわかる．

(a) 入力画像



(b) パーツ分割結果 (上) とアルファマッティングの結果 (下)



(c) パーツ分割結果 (左) とアルファマッティングの結果 (右)

図 4.7: アルファマッティングによるジャギー処理の結果.
(画像の出典: https://gamy.jp/uchi_hime/dictionaries/girls/40478)

第5章　結果と考察

5.1　LazyBrushとの比較

C++を用いて実装を行い，4.20GHzのCPUと32GBのメモリを搭載したPCで実験を行った．入力画像の解像度はおよそ1000×1000である．領域分割の結果を図5.1に示す．(a)は入力画像，(b)はユーザ入力，(c)は既存手法であるLazyBrushを用いた結果，(d)はLazyBrushに境界線入力を加えた結果，(e)はLoGフィルタを適用後，ヒストグラム平坦化を行い，さらにLoGフィルタを適用した結果である．(f)は彩度と明度を用いてLoGフィルタを適用した結果である．(g)は(e)と同様のフィルタ処理を適用した画像に境界線入力を加えた結果である．それぞれの下側の画像は上側の紫枠の部分を拡大した画像である．また，(b)の右側の画像上の紫の破線は(d)と(g)において境界線を入力した位置である．(c)と(e)では，左肩の位置にエッジがないために上腕の領域が胴の領域にまで広がってしまった．これを正すためにはユーザが色線を追加する必要があるが，LazyBrushの計算量は色線の数と画素数に比例するため，色線が増えるほど計算時間は大きくなる．色線が5個の場合には4秒ほどで計算が終わるが，色線が10個以上になると1回の入力につき計算は8秒以上かかるようになる．一方で，(d)と(g)ではユーザが境界線を指定できるためにユーザの意図した部分で領域を分割できる．領域が意図せず広がることを防ぐために色線を追加する必要がなくなり，作業時間の短縮に貢献できる．また，(c)では髪と他の部位が重なる部分で髪の輪郭線をエッジとして検出できず，髪の隙間の部分も髪の毛の領域として領域分割されている．また，左腕と髪の間の境界線もエッジとして検出できずに，髪の部分が左腕の領域となっている．(c)と比べて，(e)と(f)では髪と他の領域が重なっている部分で髪の輪郭線を正しく検出できている．(e)と(f)を比べると，(e)ではより正確に髪の輪郭線を検出できている．また，(c)と比べると，(e)と(f)では領域内のエッジに色が塗られず白くなっている領域がある．本手法では4.3.1節で示したように色の塗られた領域を膨張させてパーツに分割するため，領域内の白くなっている領域は膨張によって領域に含まれるようになり，パーツ分割に影響しない．ヒストグラム平坦化を行うことで，LoGフィルタのみでは検出できないエッジを検出できるようになった．また，明度と彩度の二つを用いることでもエッジを検出できるようになった．

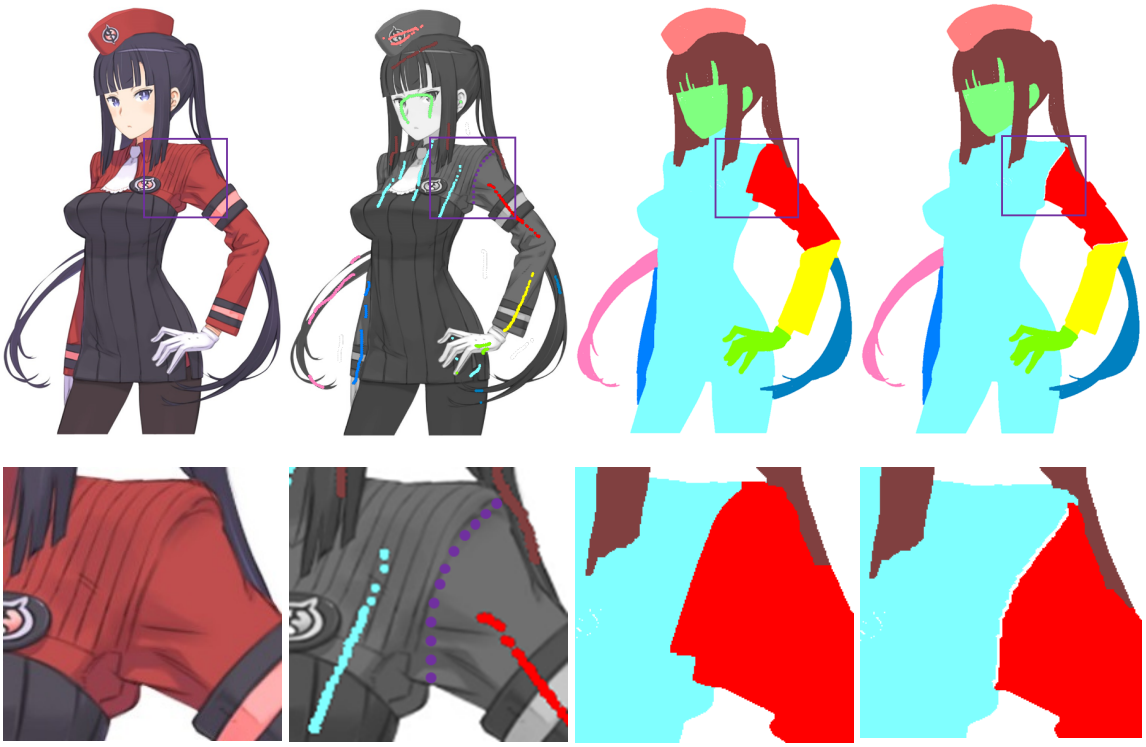
5.2　糊代の有無によるアニメーション結果の比較

境界線入力とヒストグラム平坦化を併用した手法での結果によって，糊代を付加したパーツと付加しないパーツを作成し，それぞれのパーツを用いてアニメーションを作成した．図5.2がその結果である．左手を動かすと胴の左手が重なっていた部分に穴が生じるため，Photoshopの修復ツールを用いて穴埋めした．(b)は糊代を含まない場合の結果，(c)は糊代を含む場合の結果であ

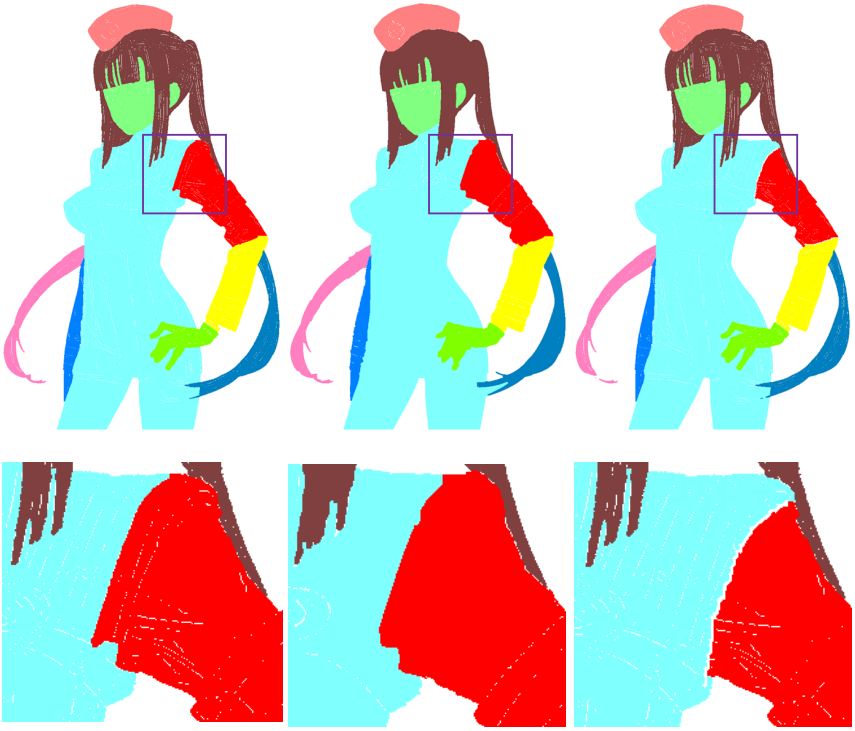
る．(b) の結果では腕を動かした後の肘の部分に隙間が生じている．一方で，(c) の結果では糊代があることにより肘の部分に隙間が生じない．

5.3 ユーザテスト

LazyBrush と提案手法の利便性を比較するため，3 名の被験者に LazyBrush と提案手法を使ってそれぞれ異なる画像の領域分割をしてもらった．被験者には数分間他の画像を用いて練習をしてから分割を行い，分割はイラストを 17 個のパーツ (髪，顔，胴，左右それぞれ上腕，前腕，手首，大腿，下腿，足) に分けるまで行う．その結果を比較が図 5.3 である．一行目，二行目，三行目それぞれで異なる被験者に実験を行ってもらっている．いずれの場合でも，作業時間は LazyBrush と提案手法で同等か提案手法のほうが短い．また，A の画像の腰と B の画像の指先は，被験者が入力を増やしても LazyBrush では背景と区別できずに色が拡がらない．一方で，提案手法ではどちらの部分も色が拡がっている．C の結果では，LazyBrush と提案手法の作業時間と分割結果はほとんど同一だが，左肩に注目すると LazyBrush では左上腕の領域が一部胴の領域にはみ出している．提案手法では境界線をユーザが指定できるために領域がはみださずに分割できる．

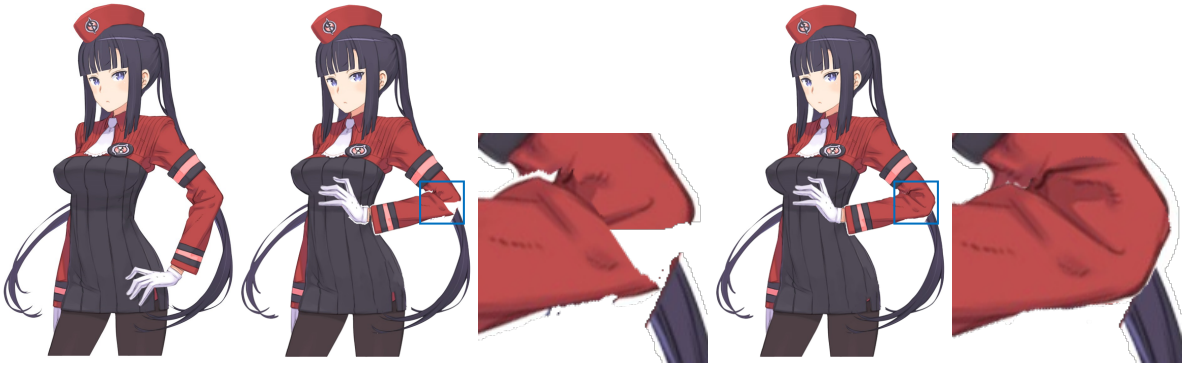


(a) 入力画像　(b) ユーザ入力　(c) LazyBrush による結果　(d) 境界線入力による結果



(e) ヒストグラム平坦化を用いた結果　(f) フィルタ処理に彩度を用いた結果　(g) (e) に境界線入力を加えた結果

図 5.1: 各手法による領域分割の結果の違い.
(画像の出典: https://cgworld.jp/interview/1306-live2d-summonnight5.html)



(a) 元画像　　(b) 糊代なしのパーツ　　(c) 糊代ありのパーツ

図 5.2: 糊代の有無によるアニメーション結果の違い.

(a) 入力画像　(b) LazyBrush の入力／結果　(c) 提案手法の入力／結果

図 5.3: LazyBrush と提案手法を用いたユーザテスト.

第6章　結論と今後の課題

6.1　結論

本研究では，関節で可動する変形モデルによるアニメーションを実現するためのイラストの領域分割をより簡単に行うために，既存手法である LazyBrush を改良した．輪郭線がエッジとして認識されなくなる問題を解決するために，既存手法で用いられるフィルタリングにヒストグラム平坦化を加えた．また，フィルタ処理に彩度を用いた．さらに，ユーザに明示的に境界線を指定させることで，エッジがない場合にも望んだ位置で領域を分割できるようにした．加えて，可動部分に糊代を追加したパーツ分割をできるようにし，アニメーションで隙間が生じないようにした．

6.2　今後の課題

第 4.3.2 節で示したように，糊代を付加したパーツ分割をする際に糊代が不要な領域をとる場合があるという問題がある．したがって，糊代の大きさを適切に選べるようにする必要がある．ユーザが糊代部分を指定する手法が考えられる．また，元々のイラストでパーツ同士が重なっていた場合，パーツ分割後に下側のパーツには空白ができてしまうので，その空白部分を埋める必要がある．これは Hui ら [7] が提案した手法によって空白領域を埋める手法を実験する．さらに，画像の解像度が大きいとユーザの入力による結果が返ってくるまでに時間がかかり，対話的なシステムとしては不便である．したがって，計算の高速化にも取り組む必要がある．

謝辞

本研究を進めるにあたり，多くのご指導ご協力をいただきました本学システム情報系の金森由博准教授，遠藤結城助教および三谷純教授に深く感謝を申し上げます．そして，日頃から研究に関する有益なご意見を多くいただきました非数値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] Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov Random Fields with Effcient Approximations. *In Proceedings of Cambridge Philosophical Society*, pages 648–655, 1998.

[2] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings of Eighth IEEE International Conference on Computer Vision (ICCV 2001)*, volume 1, pages 105–112. IEEE, 2001.

[3] Rother Carsten, Kolmogorov Vladimira, and Blake Andrew. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004.

[4] Yung-Yu Chuang, Brian Curless, David H Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, volume 2, pages II:264–II:271. IEEE, 2001.

[5] Yuki Endo, Satoshi Iizuka, Yoshihiro Kanamori, and Jun Mitani. DeepProp: Extracting deep features from a single image for edit propagation. *Computer Graphics Forum*, 35(2):189–201, 2016.

[6] Mark Everingham, Chris J. Needham, and Roberto Fraile, editors. *Proceedings of the British Machine Vision Conference 2008, Leeds, September 2008*. British Machine Vision Association, 2008.

[7] Huang Hui, Yin Kangxue, Gong Minglun, Lischinski Dani, Cohen-Or Daniel, Ascher Uri, and Chen Baoquan. "Mind the Gap": Tele-Registration for Structure-Driven Image Completion. *ACM Transactions on Graphics (TOG)*, 32(6):174:1–174:10, 2013.

[8] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Transactions on Graphics (TOG)*, 23(3):689–694, 2004.

[9] Eric N Mortensen and William A Barrett. Intelligent scissors for image composition. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 191–198. ACM, 1995.

[10] Renfrey Burnard Potts. Some generalized order-disorder transformations. *In Proceedings of Cambridge Philosophical Society*, 48(1):106–109, 1952.

[11] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. Manga colorization. *ACM Transactions on Graphics (TOG)*, 25(3):1214–1220, 2006.

[12] Mark A Ruzon and Carlo Tomasi. Alpha estimation in natural images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000)*, volume 1, pages 18–25. IEEE, 2000.

[13] Daniel Sýkora, Jan Buriánek, and Jiří Žára. Colorization of black-and-white cartoons. *Image and Vision Computing*, 23(9):767–782, 2005.

[14] Daniel Sỳkora, Jan Buriánek, and Jiří Žára. Unsupervised colorization of black-and-white cartoons. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering (NPAR 2004)*, pages 121–127. ACM, 2004.

[15] Daniel Sýkora, John Dingliana, and Steven Collins. LazyBrush: Flexible Painting Tool for Hand-drawn Cartoons. *Computer Graphics Forum*, 28(2):599–608, 2009.

[16] Richard Zhang, Jun yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, and Alexei A. Efros. Real-time User-guided Image Colorization with Learned Deep Priors. *ACM Trans. Graph. (TOG)*, 36(4):119:1–119:11, 2017.