Smart Layer Splitter：pix2pix を用いたデジタルイラスト制作の色塗り工程における自動レイヤ分けシステム

渡邉優[1]　　阿倍博信[1]

概要：デジタルイラスト制作の一工程である色塗り工程では，線画を髪や肌などのパーツごとにレイヤ分けする作業が必要である．しかし，既存のグラフィックソフトに付属する塗りつぶしツールでは，手作業のため手間がかかってしまうという問題があった．そこで，本論文では，conditional GAN の一方式である pix2pix を用いてレイヤ分け作業を自動化する方式について提案する．さらに，提案方式に基づく自動レイヤ分け処理において誤りが発生した場合でも誤りを手動で修正する UI を持った自動レイヤ分けシステム:Smart Layer Splitter を開発し，その有効性について評価を行った. その結果, 既存のグラフィックソフトと比較して, 作業時間を 39.8%短縮できるとともに, 操作回数を 68.6%削減できることが確認でき，システムの有効性について確認することができた.

Smart Layer Splitter: An Automatic Layer Splitting System for Coloring Process of Creating Digital Illustration using pix2pix

YU WATANABE[1]　　HIRONOBU ABE[1]

1. はじめに

一般的に, デジタルイラストの制作工程は, ラフ, 線画, 色塗りの 3 つの工程から構成され，さらに色塗り工程はレイヤ分けと影塗りの 2 つの工程から構成される．レイヤ分けは線画に対して，肌や髪などのパーツごとにレイヤに分ける作業であり，後で行う影塗りの下地となる部分を作成する工程である．

従来，レイヤ分け作業は，グラフィックソフトに付属する塗りつぶしツールや選択範囲ツールを使用して行う場合が多く，選択したい領域の先端が細い場合や，線画の線が途切れている部分に対してこれらのツールを使用した場合, 領域から色があふれ出してしまい，ユーザの想定どおりに選択することができない．その対応として人手で線画の隙間を修正した後，再度塗りつぶしツールを使用する必要があるため，修正作業に手間がかかる問題がある．さらに，このレイヤ分け作業は領域を単色で塗りつぶす単純作業であり，イラストの品質に直接影響しない．このような背景から, レイヤ分け作業を AI 技術により自動化することで, イラストレータのデジタルイラスト制作作業の支援になると考えられる．

本論文では，デジタルイラストの制作現場において，AI 技術を用いたレイヤ分け作業の自動化を目的として，conditional GAN の一種である pix2pix[1]を用いてレイヤ分け作業を自動化する方式について提案し，それを用いた自動レイヤ分けシステムである Smart Layer Splitter の開発及びその評価について述べる．

以下，本論文では，2 章にて関連研究，3 章にて自動レイヤ分け方式の提案, 4 章にて自動レイヤ分けシステム:Smart Layer Splitter の開発，5 章にてシステム評価，6 章にて考察について述べ，最後にまとめを行う．

2. 関連研究

前述のとおり，デジタルイラストの制作工程は，ラフ，線画，色塗りの 3 つの工程から構成される．この中でも，本論文でも対象としている色塗り作業への AI 技術の適用は最も研究が盛んな分野の一つである．

PaintsChainer[2], style2paints[3]は, ユーザが線画をアップロードし，領域の色や影を指定することで色塗りを自動で行う Web サービスである. PaintsChainer の出力画像は 2020 年 5 月時点で 3 種類の中からユーザが指定することができる．一方で style2paints はユーザが参照画像を入力すると，その参照画像の色を反映させて色を塗ることができ，それに加えてライティング画像，線画無しの色塗り画像などを選択し出力することができる．

また，漫画の着色に関する研究も存在する．Comicolorization[4]は漫画の自動彩色システムである．モデルは白黒写真に対して自動的に色塗りを行う CNN を漫画に適応したものである．システムに対して，線画と一緒に参照画像を入力すると，参照画像の色を使用して線画に色

1 東京電機大学
Tokyo Denki University

を塗る．

また，セグメンテーションを使用して漫画の着色を行う研究も存在する[5]．画像処理を用いて漫画のトーンを除去した後，線画と参照画像を pix2pix に入力して着色を行い，セグメンテーションや色の補正によってグレースケールの漫画をカラー漫画に変換している．本研究では，セグメンテーションについての手法を参考にしている．

上記以外にも，AI 技術を活用して色塗り作業を自動化する研究が行われているが，基本的に色塗り作業の全工程を自動化する技術が中心であるため，出力画像が 1 枚のみであり，本論文で必要となるパーツごとのセグメンテーションや，レイヤ分けが行われていないため，本論文の目的には適さない．

デジタルイラスト制作以外の分野では，セマンティックセグメンテーション技術の一つである U-net[6]を用いてアニメ制作における色塗りの自動化に関する研究も存在する[7]．アニメは同じキャラクタが複数出現するため，キャラクタごとに U-net を用いて学習させることで，色塗り作業の自動化を実現している．また，[5]と同様に，色塗りの前段階で精度を下げる要因となる情報の削除や塗りたい領域をはみ出して色塗りがなされてしまった場合，セマンティックセグメンテーションなどを組み合わせて補正することで，実行結果の改善を図っている．

また，色塗り作業以外の作業に対する制作支援としては，線画作業への AI 技術の適用に関する研究も存在する．線画の作成作業は，ラフと呼ばれる大雑把な構図を描いた物から適切な線を選択し，なぞる作業であり，smartInker[8]では，GAN を用いることで，ユーザはラフの画像を入力し，消去したい線と加筆したい線を指定することで線画の自動生成を実現している．

3. 自動レイヤ分け方式の提案

3.1 基本方針

2 章の結果を踏まえ，本論文にて提案する自動レイヤ分け方式の基本方針について下記のとおり整理する．

- 先行研究[5][7]でも採用しているセマンティックセグメンテーション技術を用いて，レイヤ分けに必要なパーツごとの自動セグメンテーションを実現する．
- 自動化処理結果が誤った場合を想定し，後処理としての画像補正処理を組み合わせ，精度向上を図る．
- 後処理としての画像補正処理で補正しきれなかった箇所については，手動で修正する．

3.2 自動レイヤ分け方式の概要

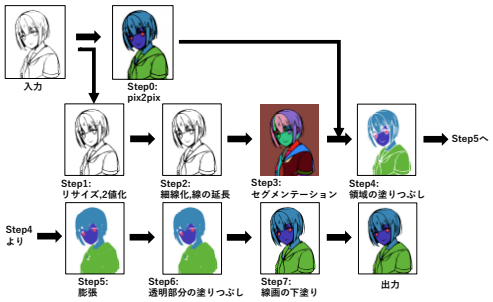

図 1　pix2pix を用いた自動レイヤ分け方式の概要

図 1 に本論文にて提案する pix2pix を用いた自動レイヤ分け方式の概要について示す[9]，[10]．最初にユーザが入力した線画を，conditional GAN の一種である pix2pix を用いて，髪，肌，目，服の 4 つのパーツに分ける形で色を塗る．このとき pix2pix の実行結果は，色のはみだしやにじみが存在する可能性が高く，そのままレイヤ分け画像として使用することはできない．レイヤ分けとは，パーツの領域を単色で塗りつぶす作業であり，領域から色がはみだすことや，色の不安定な場所があってはならない．例えば肌の領域が不安定な場合，背景の色が透ける可能性があり，完成作品に悪影響を及ぼす可能性があるためである．そこで，pix2pix の実行結果に対する後処理として，線を塞ぐ処理とセグメンテーションを中心とする画像補正処理を行い，その結果をレイヤに分割し，最終出力とする．以下，pix2pix とその後処理について，処理内容の詳細について説明する．

3.3 入力画像の仕様

今回対象とする入力画像は，キャラクタイラスト制作を想定したキャラクタの線画とする．以下に入力画像の仕様について示す．

- 画像データの想定仕様

線は背景と十分に明度が違うものとし，線は輝度が 200 cd/m^2 以下であり，背景は 200 cd/m^2 以上である必要がある．また，画像内にノイズのないものとし，スマートフォンのカメラで写真を撮ったような影の映り込みはないものを想定している．また，背景は含まず，キャラクタのみが写っていることを前提条件とする．画像サイズは指定しないが，幅，高さとも 100px 以上 2,000px 以下を目安とする．

- キャラクタの仕様

画像に上半身のみが映っているもの対象とし，2 頭身程度のデフォルメされたキャラクタは対象外とする．髪の長さや髪形については制限せず自由とする．目の形状や大きさについても制限せず，目を閉じているものについても良いとする．顔の大きさについても制限しないが，顔全体が画面に写っているものとし，見切れていないものとする．また，顔は正面に近いものとし，横向きや後ろ向きの画像は対象外とする．

- 線の仕様

線画において，線の太さやテクスチャといったスタイル

は自由とする．髪などの一部の領域や影をベタで塗りつぶしている画像を含んでいても良いが，斜線やトーンは含まないとする．また，ラフのような跡切れが大きいものや，髪の毛において毛束の先端の線を閉じていないもの，線の隙間を故意に作っている絵については，線の補間が難しいため対象外とする．

3.4 出力画像の仕様

出力画像は服，髪，肌，目，背景の 5 つのパーツと線画 1 枚がセットになった PSD (PhotoShop Document)[11]形式とする．出力画像の概要について図 2 に示す．

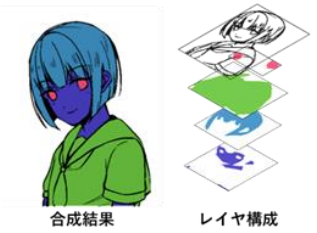

図 2　出力画像の概要

PSD 形式は一般的なグラフィックソフトで扱うことが可能であり，レイヤ構造を格納することができる．レイヤは上から線画，目，服，髪，肌，背景の順とし，すべてのレイヤの合成モードは通常モードとする．また，ほかのパーツ領域と被らないようにする．例えば，目のパーツの領域は肌パーツの目の領域部分については塗られておらず透明とする．また，線の下について，線のスタイルや透明度によっては背景色が透けてしまうため，線の周囲のパーツ色を塗るようにする．パーツの間の線については線の中心線を境界とする．

3.5 Pix2pix を用いた自動レイヤ分け処理

(1) pix2pix の概要

pix2pix とは Isola らによって考案された conditional GAN を用いて Image to Image のタスクを汎用的に行うアルゴリズムである．GAN とは生成側(Generator)と判断側(Discriminator)の 2 つのモデルから構成されるモデルである．pix2pix では Generator として先行研究[7]においても採用している U-net を採用している．U-net は医療画像のセグメンテーションにおいて高スコアを持つモデルであり，入力と出力は画像である．そのため，pix2pix は画像から画像へ高品質で変換を行うことができる．図 3 に pix2pix のブロック構成について示す．

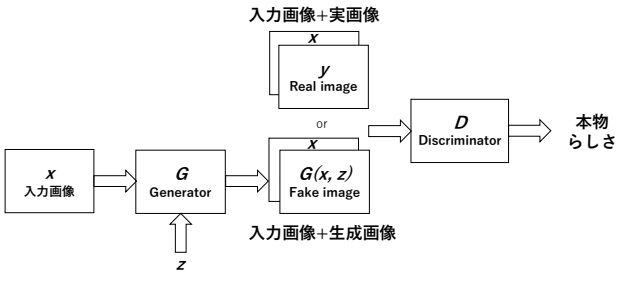

図 3　出力画像の概要

(2) 学習データの概要

イラストコミュニケーションサービス pixiv[12]から，入力画像の想定仕様を満たす線画を 345 枚収集した．収集した画像サイズは平均で幅：662px，高さ：713px であった．次に前処理として，収集した線画を肌，髪，目，服の 4 つのパーツをそれぞれ別の色で塗り分けを行った．このとき，塗り分けはアンチエイリアスなしで行っている．また，背景を白で塗りつぶし，RGB の 3 チャンネルとした．作成したデータを 9:1 に分け，9 割を学習用データ，残りの 1 割をテストデータとした．

次に，水増し処理として左右反転，1〜4 度の回転，0.5〜1.5 倍にリサイズ処理を行った．また，線画とレイヤ分け画像について，ニアレストネイバー法を用いてリサイズした．このとき，アスペクト比を保持したまま 256px×256px にリサイズするとともに，アスペクト比の変更によって生じた空白部分は黒で塗りつぶした．ニアレストネイバー法を用いた理由は色数を増やさないためである．水増し処理を行った結果，作成したデータは学習データ:12,420 枚，テストデータ:1,380 枚となった．作成した学習用データの例について図 4 に示す．

図 4　学習用データの例

(3) pix2pix によるモデル作成

次に，pix2pix による学習によるモデル作成について述べる．開発環境としては，ディープラーニングのフレームワークとして chainer[13]を選択し，chainer-pix2pix[14]の環境を構築し，Python を用いて pix2pix のモデル作成を行った．また，学習のパラメータは batchsize を 2 とし，patchSize を 256px とした．

Generator 側については，入力画像を 256px×256px の線画とし，教師ラベルを 256px×256px のレイヤ分け画像とした．また，入力チャンネル，出力チャンネルは 3 とした．損失関数は教師ラベルと Generator の出力結果の平均絶対誤差×100 + adversarial loss とした．

Discriminator 側について，入力画像は Generator の 256px×256px の生成画像と教師ラベルとした．また，入力チャンネルを 3，出力チャンネルを 1 とした．損失関数は adversarial loss とした．最適化方式は Adam とし，パラメータは α=0.0002，β1=0.5，β2=0.999 とした．

図 5 に，テストデータを使用した際の損失関数の出力の推移を示す．その結果，Generator 側の損失関数の出力は 250 から 25 まで低下した．また，Discriminator は 0.5 付近を往復していることから，正常動作していることを確認できた．80,000 iterator で損失関数が最小となったと判断し，このと

きのモデルを pix2pix2 のモデルとして採用した．

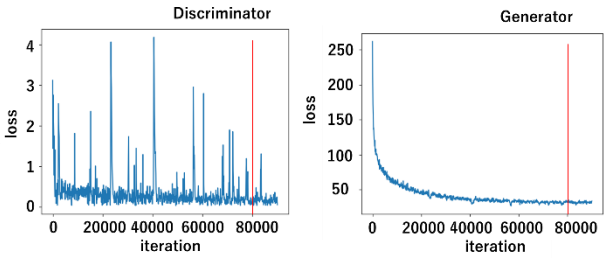

図 5　損失関数の出力の推移

3.6 後処理としての画像補正処理

前節で作成した pix2pix のモデルに対して，線画を入力した際の出力結果を図 6 に示す．

図 6　pix2pix の出力

図 6 を見ても分かるとおり，pix2pix の出力結果は同一の領域内が単色ではない．レイヤ分け画像はすべての領域が単色とする必要がある．前節で説明したとおり，レイヤ分けとはパーツの領域を単色で塗りつぶす作業であり，領域からはみ出すことや色の不安定な場所があってはならないためである．以上の理由から，pix2pix の出力結果に対して後処理として画像補正処理を行うこととした．

後処理の基本方針について下記のとおり整理する．

- 方針としては先行研究[5][7]と同様に，セグメンテーションを使用して後処理を行う．
- セグメンテーションの際，イラストの線が完全に塞がれているとは限らない．線が塞がれていない場合は，精度が低下する可能性がある．そのため，線画の跡切れを検出し，延長することで線の途切れを減らす．
- pix2pix の出力結果にセグメンテーションを適応した際に領域が小さい場合，領域内のにじみの影響を受けてしまい，色を検出できない可能性がある．小さい領域は周辺のパーツに属する可能性が高いため，周辺ピクセルを参照して領域をパーツに分類する．
- 線画の透明度によっては，線の下まで塗りつぶさなければ背景が透けてしまう可能性がある．そのため，各パーツについて領域の膨張を行うことで線画の下まで塗りつぶす．

以下，図 1 の提案方式概要で示した Step ごとに，後処理の内容について説明する．処理速度の観点から Step2 のみ C 言語で実装し，他の Step はすべて Python で実装した．

(1) Step1:リサイズ，2 値化

pix2pix の出力サイズを線画のサイズに合わせてリサイズする．pix2pix の出力画像は 256px×256px であるため，元の線画のサイズにリサイズ処理を行う．また，入力線画を 2 値化する．Step2 にて細線化を適応する必要があるためである．2 値化する際，輝度が 200 cd/m^2 以下を黒(線)，それ以上を白(背景)とし，背景は透明色に置き換える．

(2) Step2:細線化，線の延長

2 値化した入力線画を Nagendraprasad-Wang-Gupta のアルゴリズム[14]で細線化を行う．細線化画像から端点を検出し，隣接する線を 5px 辿り，コピーする．コピーした線を端点に貼り付け，線の方向に対して延長を行う(図 7).

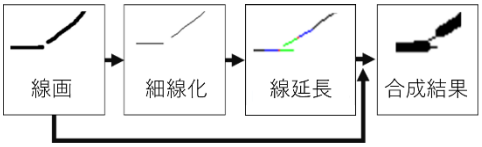

図 7　線の延長

これによって線の跡切れを軽減し，Step3 で行うセグメンテーションの精度を高める．端点とは 8 近傍のピクセルのうち，1px が黒であり，そのほかは透明である点と定義した．図 7 において，赤色のピクセルは検出された端点，青色のピクセルは端点から 5px 辿った線，緑色のピクセルが延長した線である．

(3) Step3:セグメンテーション

Step2 の結果と Step1 の線画を結合し，その結果について，閉領域の塗りつぶしを行い，すべての領域に対してセグメンテーションを行う．

(4) Step4:領域の塗りつぶし

Step0 の pix2pix の出力と Step3 の出力を参照し，各領域に最も含まれているパーツの色を用いて領域を塗りつぶす．これをすべての領域に対して行う．このとき，領域を矩形として面積が 500px 以下となった場合は，領域が小さいと判断し，塗りつぶさない．小さい領域については pix2pix の出力を参照した際，にじみに大きく影響される可能性があり，パーツの分類に悪影響を及ぼすためである．このとき，各色は異なるレイヤとして描画する．

(5) Step5:膨張

レイヤのパーツの各色について 3px の膨張を行う．これによってパーツの間の線の下を塗りつぶす．線画の透明度によっては，線の下まで塗りつぶさなければ背景が透けてしまう可能性があるためである．このとき，線画の黒色領域をマスクとし，線画の領域のみに色が乗るようにする．

(6) Step6:透明部分の塗りつぶし

Step4 にて塗りつぶされなかったピクセルについて，8 近傍の色を参照して，最も多かった色を取得し，その領域に閉領域の塗りつぶしを適応する．ただし，周辺ピクセルに透明色が一番多かった場合は塗りつぶさない．また，領域が背景色であると判断された場合には，参照するピクセルを 15 近傍に増やしてもう一度スキャンを行い，その結果の色を塗る．

(7) Step7:線画の下塗り

線画と延長線の下は色が塗られていないピクセルが存在するため，塗られていないピクセルについて，8 近傍のピクセルの色をカウントし，最も多かった色を線画の下に塗る．透明色のピクセルがなくなるまで Step7 を繰り返す．

3.7 自動レイヤ分け方式の精度評価

セマンティックセグメンテーションの評価方式の一つである Mean Accuracy[16]を用いて，提案した自動レイヤ分け方式の精度評価を行った．Mean Accuracy とは，パーツごとに精度を求める評価方式の一方式であり，パーツの面積に精度が依存しないことが特徴である．今回使用した Mean Accuracy の定義について式(1)，(2)に示す．

$$Mean\ Accuracy = \frac{1}{n_{cl}} \cdot \sum_i \frac{n_{ii}}{t_i} \tag{1}$$

$$t_i = \sum_j n_{ij} \tag{2}$$

ここで，n_{ij} はクラス j に属すると予測されるクラス i に属するピクセル数，n_{cl} はクラス数，t_i はクラス i の総ピクセル数とする．また，pix2pix の出力形式は RGB であるため，各ピクセルを使用したパーツのパレットの色に変換する形で評価を行った．変換のルールはパレットの色で一番近い色を採用する方式を選択した．

表 1 に，自動レイヤ分け方式の精度評価の結果について示す．

表 1　自動レイヤ分け方式の精度評価の結果

	平均精度 (%)				
	肌	髪	服	目	全体
pix2pix	74.5	87.5	84.4	56.0	81.6
pix2pix+後処理	78.6	90.0	86.0	64.2	84.8

表 1 の結果から，pix2pix を用いた自動レイヤ分け方式の全体の精度が 81.6%となることが確認できた．その後，後処理により，全てのパーツにおいて精度が向上していることを確認することができた．また，後処理後の全体の精度として 84.8%となることが確認できた．

3.8 自動レイヤ分け処理結果の考察

後処理により，全てのパーツにおいて自動レイヤ分けの精度が向上していることを確認することができ，全体の精度として 84.8%となることが確認できた．また，パーツごとの精度では，後処理後で最も精度が高かったのが髪で 90.0%，低かったのが目で 64.2%となった．しかし，目は後処理によって 7.8%と最も精度が向上していることが確認できた．

提案した後処理によって精度が上がった理由として，一部の色が不安定であった領域が単色になったために，本来のレイヤ分け画像の仕様に基づいたことが要因であると考えられる．また，目の領域について精度が大きく上昇した要因として，領域が小さくパーツの数も少ないことから，後処理の影響を大きく受けたことでこのような結果になったと考えられる．

次に，自動レイヤ分け処理に失敗した箇所について原因分析を行ったところ，大きく以下の 2 種類の誤りが存在することが分かった．その分析結果について図 8 に示す．

図 8　自動レイヤ分け結果の誤り

(1) パーツの間違い

パーツの間違いとは，ある閉領域の色がパーツと対応していない場合である．図 8 の例では首が肌パーツであるにもかかわらず，服パーツであると認識されている．

(2) 色のあふれ

色のあふれは，入力線画において線が大幅に途切れている場合について，領域外まで色がはみ出した場合である．発生原因としては，大幅な線の跡切れによって，自動レイヤ分け方式の step2 において隙間を塞ぎきれず，step3 において閉領域の認識に失敗したためと考えられる．

4. 自動レイヤ分けシステム:Smart Layer Splitter の開発

4.1 開発方針

本章では，3 章で提案した自動レイヤ分け方式を適用した自動レイヤ分けシステムである Smart Layer Splitter の開発について述べる[17][18]. システムの開発にあたり, 3.1 の基本方針に従い，自動レイヤ分け処理で発生した誤り部分については，手動で修正する方針とする．また，今回修正の対象とする自動レイヤ分け処理の誤りは，パーツの間違いのみとする．パーツの間違いは領域の色を修正するのみで対応できるためである. レイヤ分けに対応するパーツは，提案方式に従い，肌，髪，服，目，背景の 5 種類とする．

4.2 システム構成

本システムはPC と液晶ペンタブレットから構成される．本システムで使用した PC のスペックについて表 2 に示す．

表 2　PC のスペック

プロセッサ	intel Core i7-7500U
システムクロック	2.70 GHz
RAM	8,192 MB
ビデオカード	NVIDIA GeForce 940MX 4GB
OS	Windows 10 PRO 64bit

また，液晶ペンタブレットは Huion 社の kamvas PRO 12 を採用した．画面のタッチ機能はついておらず，ペンにはサイドにボタンが付属している．開発したシステムの外観について図 9 に示す．

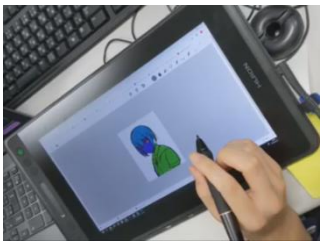

図 9　システムの外観

次に，開発したシステムのソフトウェア構成について図 10 に示す．本システムは Vue.js と HTML5，JavaScript，jQuery を用いて SPA(Single Page Application)形式の Web アプリケーションとして構築した．画面のスタイルは Bootstrap を，Web サーバは Python のライブラリである Flask を用いて構築した．自動レイヤ分け処理は，Python と C を用いて，線画を自動的にレイヤ分け画像に変換する処理を実装した．

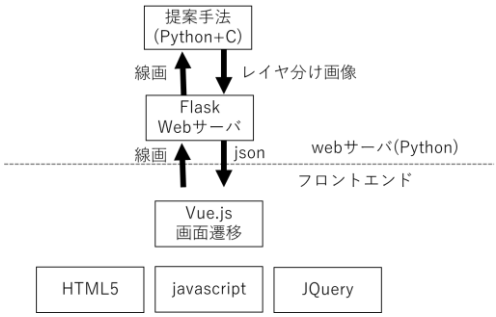

図 10　ソフトウェア構成

4.3　レイヤ分け処理の流れ

本システムを用いたレイヤ分け処理の流れを図 11 に示す．ユーザはブラウザを用いて，レイヤ分けを行う線画を選択し，ユーザ操作によりサーバにアップロードする(図 11 ①)．次に，サーバで自動レイヤ分け処理を実行する(図 11 ②)．最後に，サーバは自動レイヤ分けの処理結果をブラウザ側に戻す．処理結果に，パーツの誤りなどの修正すべき誤りが存在する場合は，ユーザがブラウザ上で修正を行う(図 11③)．

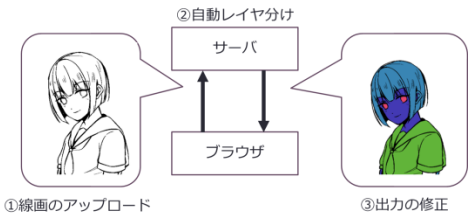

図 11　レイヤ分け処理の流れ

4.4　UI の概要

本システムの UI の概要を図 12 に示す．UI 上部のツールバーは，ブラシの変更や色の変更を行うことができる．ツールバーのアイコンは，左から①PSD 出力，②拡大/縮小/アンドゥ，③色の選択/ツールの変更/ブラシサイズの変更である．以下，UI の構成要素の詳細について説明する．

図 12　UI の概要

(1)　PSD 出力

自動レイヤ分け結果の修正が終わった後，ツールバー左上のレイヤ分け画像出力ボタンをクリックすることで，編集結果がサーバに送信され，出力結果として，修正結果を反映した PSD 形式のレイヤ分け画像を得ることができる．

(2)　拡大/縮小/アンドゥ

拡大と縮小は，キャンバスの表示サイズを変更することができる．拡大することで，より細かい部分の修正が可能となる．また，アンドゥは編集結果を一つ前に戻すことができる．

(3)　色の選択/ツールの変更/ブラシサイズの変更

色の選択アイコンをクリックすると，色の選択ウィンドウが表示される．色の変更ウィンドウでは，パーツの色一覧を表示する．色の上にカーソルを乗せると，その色に対応するパーツ名が表示される．色をクリックすることで，選択した色を描画色に変更することができる．

ツールの変更では，左から，手のひらツール，塗りつぶしツール，投げ縄選択ツール，矩形選択ツール，ペン選択ツール，ペンツールが配置されている．また，ペン選択ツールとペンツールを選択している場合のみ，隣にブラシサイズの変更バーが表示され，ブラシのサイズの変更を行うことができる．

4.5　ツールの概要

(1)　手のひらツール

手のひらツールを選択すると，ドラッグ操作によりキャンバスを上下左右に動かすことができる．

(2)　塗りつぶしツール

指定された 1 つの領域の内部を塗りつぶすツールであり，背景など比較的大きな領域を塗りつぶす場合に使用する．塗りつぶしツールの使用方法を図 13 に示す．キャンバス画面で，塗りつぶしを行う領域内をクリックすることで線の中の閉領域を塗りつぶすことができる．

図 13　塗りつぶしツールの使用方法

(3) 投げ縄選択ツール

キャンバス上で修正する領域を，投げ縄選択ツールを用いて，ドラッグ操作により自由形式で描画すると，カーソルの軌道から線の中の閉領域を認識し，閉領域の内部を塗りつぶすことができる．

(4) 矩形選択ツール

キャンバス上で修正する領域を，投げ縄選択ツールと同様矩形選択ツールを用いて，頂点を指定する形でクリック操作により矩形を描画すると，矩形領域の中の閉領域を認識し，閉領域の内部を塗りつぶすことができる．

(5) ペン選択ツール

耳や髪の隙間などの小さい複数の閉領域を修正する際に使用し，キャンバス上の修正する領域を，ペン選択ツールを用いて描画すると，線の中の閉領域を認識し，内部を塗りつぶす．ペン選択ツールの使用方法について図 14 に示す．

図 14　ペン選択ツールの使用方法

(6) ペンツール

一般的なグラフィックソフトのブラシツールに相当するツールである．選択ツールは閉領域しか塗ることができないため，修正したい領域が閉領域ではない場合に使用するツールである．修正する部分の色を色選択ウィンドウから選択し，キャンバス上でカーソルをドラッグすることで，線を描くことができる．他のツールとは異なり，閉領域の認識は行わない．

5. システム評価

5.1 評価方法

開発した自動レイヤ分けシステムの有効性の確認を目的としたシステム評価を実施した．具体的には，実験参加者として大学生 23 人に対して，本システムと既存のグラフィックソフトを用いて実際にレイヤ分け作業を行う評価実験を実施し，作業時間と操作回数を計測するとともに，アンケートを用いたユーザビリティの評価を行った．

また，ユーザに対してアンケートを行い，日常的にデジタルイラストを描くと回答した実験協力者 7 人を経験者，それ以外の 16 人を初心者と定義し，経験別に作業時間と操作回数の計測とユーザビリティ評価を行った．

評価実験での課題を図 15 に示す．評価実験で使用した線画は，図 15 の入力線画であり，サイズは 416px×512px，アンチエイリアスなしとした．アンチエイリアスなしとした理由は，アンチエイリアスありの線画は本システム，既存のグラフィックソフトともにレイヤ分け作業が煩雑になるからである．また，本システムを使用し，自動レイヤ分け後に修正の必要となる領域は 7 箇所であった．

図 15　評価実験での課題

5.2 作業時間の評価

本システムと既存のグラフィックソフトを用いてレイヤ分け作業を行った際に計測した作業時間について図 16 に示す．また，経験別の作業時間について図 17 に示す．

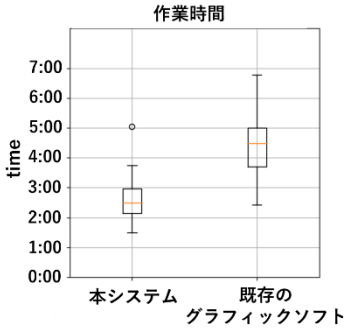

図 16　作業時間の評価結果

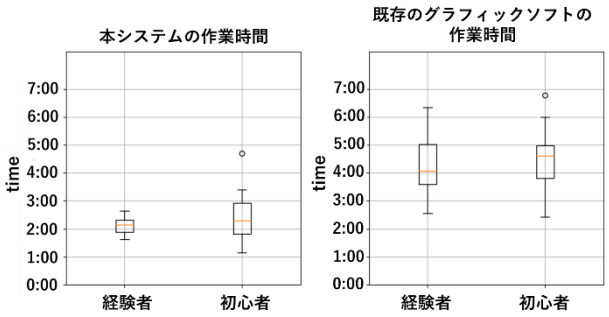

図 17　経験別の作業時間の評価結果

5.3 操作回数の評価

本システムと既存のグラフィックソフトを用いてレイヤ分け作業を行った際に計測した操作回数について図 18 に示す．また，経験別の操作回数について図 19 に示す．

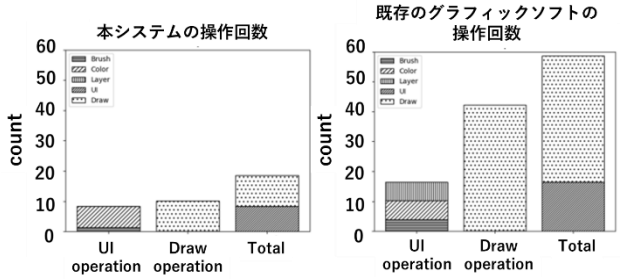

図 18　操作回数の評価結果

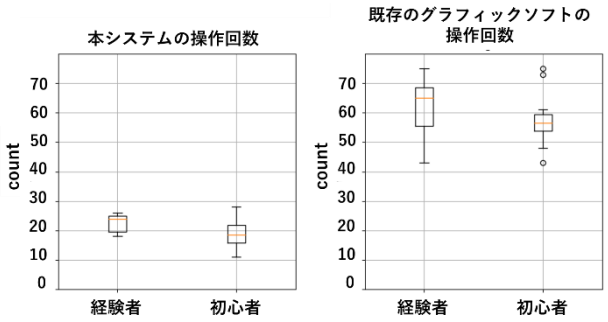

図 19　経験別の操作回数の評価結果

5.4 ユーザビリティの評価

アンケートを用いたユーザビリティの評価結果について表 3 に示す．また，作業時間と操作回数と同様に，アンケートの結果をもとに経験者と初心者に分けた際の評価結果をそれぞれ表 4，表 5 に示す．

表 3　ユーザビリティの評価結果

アンケート内容	結果
最後までレイヤ分けを行うことができましたか？	**はい(23)** いいえ(0)
システムは使いやすいですか？	5(6) **4(14)** 3(2) 2 1
システムの操作に迷うことがありましたか？	はい(9) **いいえ(14)**
自動で塗り分けした結果について満足ですか？	**はい(18)** いいえ(5)
実際にあったらイラスト制作の一部で使用したいですか？	**はい(22)** いいえ(1)
ボタンの位置やツールバーの位置は適切でしたか？	**適切(14)** その他(9)

表 4　経験者の評価結果

アンケート内容	結果
最後までレイヤ分けを行うことができましたか？	**はい(7)** いいえ(0)
システムは使いやすいですか？	5(1) **4(4)** 3(2) 2 1
システムの操作に迷うことがありましたか？	はい(3) **いいえ(4)**
自動で塗り分けした結果について満足ですか？	**はい(4)** いいえ(3)
実際にあったらイラスト制作の一部で使用したいですか？	**はい(6)** いいえ(1)
ボタンの位置やツールバーの位置は適切でしたか？	適切(3) **その他(4)**

表 5　初心者の評価結果

アンケート内容	結果
最後までレイヤ分けを行うことができましたか？	**はい(16)** いいえ(0)
システムは使いやすいですか？	5(5) **4(11)** 3 2 1
システムの操作に迷うことがありましたか？	はい(6) **いいえ(10)**
自動で塗り分けした結果について満足ですか？	**はい(14)** いいえ(2)
実際にあったらイラスト制作の一部で使用したいですか？	**はい(16)** いいえ(0)
ボタンの位置やツールバーの位置は適切でしたか？	**適切(11)** その他(5)

6. 考察

6.1 作業時間の評価に関する考察

(1) 実験協力者全体の考察

図 16 において，作業時間の平均値は，本システム : 2 分 31 秒，既存のグラフィックソフト : 4 分 44 秒となり，作業時間を 46.8%短縮することができた．また，自動レイヤ分け処理時間の 10 回計測時の平均は 20.2 秒であり，自動レイヤ分け処理時間を加算した結果は, 2 分 51 秒となるため，自動処理の時間を加えても既存のグラフィックソフトよりも作業時間を 39.8%短縮できることを確認した．これは本システムでは誤りのみ修正すれば良い点と，レイヤと色の選択を同時に行うことができることが理由であると考える．

(2) 作業経験に関する考察

図 18 において，経験者の作業時間の平均値は，本システム : 2 分 16 秒，既存のグラフィックソフト : 4 分 18 秒となり，初心者は本システム : 2 分 24 秒既存のグラフィックソフト : 4 分 30 秒となった．本システムを使用した場合は，経験者が初心者よりも作業時間が 8 秒少なく，既存のグラフィックソフトを使用してした場合には，経験者が初心者よりも作業時間が 12 秒少なくなる結果となった．考えられる要因としては，経験者はグラフィックソフトや液晶タブレットの扱いに慣れていたことが考えられる．

6.2 操作回数の評価に関する考察

(1) 実験協力者全体の考察

図 17 において操作回数の平均値は, 本システム : 18.5 回，既存のグラフィックソフト : 58.9 回となり，操作回数を 68.6%と大幅に削減することができた．各項目の内訳は，ブラシや色の変更を含む UI 操作について，本システム : 8.3 回，既存のグラフィックソフト : 16.4 回となり，49.4%削減することができた．また，描画操作について，本システム : 10.2 回，既存のグラフィックソフト : 42.2 回となり，75.9%削減することができた．これは，本システムでは自動レイヤ分け処理後の修正のみ行えば良い点と，レイヤの選択と色の選択の機能を統合した点が理由であると考える．

(2) 作業経験に関する考察

図 19 において本システムの操作回数の平均値は，経験者 : 19.4 回，初心者 : 18.1 回となった．各項目の内訳は，経験者について UI 操作 : 8.0 回，描画操作 : 11.4 回となった．初心者については UI 操作 : 8.4 回，描画操作 : 9.6 回となった．

また既存のグラフィックソフトの操作回数の平均値は，経験者 : 61.6 回，初心者 : 57.3 回となった．各項目の内訳は，経験者について UI 操作 : 16.9 回，描画操作 : 44.7 回となった．初心者については UI 操作 : 16.2 回，描画操作 : 41.1 回となった．

既存のグラフィックソフトを使用した場合，経験者は初心者よりも操作回数が多い傾向が見られた．特に描画操作

については経験者の方が 3.6 回多い結果になった．この結果について，実験協力者がレイヤ分けを間違えた際に，修正する方法の違いによって，この差が現れたのではないかと考えられる．例えば，ある経験者はレイヤ分けのミスを発見した際に，消しゴムツールで間違った部分を消した後，レイヤを変更して再び色を塗る処理を行った．一方で初心者についてはレイヤ分けを間違えた際，本来そのレイヤに塗ってはならないパーツの色を置いて修正するといった行動が見られた．そのため経験者は初心者よりも操作回数が多くなる傾向が見られたと考えられる．

本システムでは，レイヤの選択と色の選択機能は統合しているため，上記のような問題は起こらない．それによって，経験者と初心者の操作回数は既存のグラフィックソフトよりも差が少なくなったのではないかと考える．

6.3 ユーザビリティの評価に関する考察

(1) 実験協力者全体の考察

表 3 において，本システムを使用した実験参加者全員が最後までレイヤ分けを行うことができた．また，使いやすさについても 5 段階評価で 4 という良好な結果が得られた．コメントでは，レイヤを切り替えしなくて良いので便利，ツールの持ち替えが楽といったポジティブなコメントを得ることができた．その一方，選択ペンでは，目的の部分が塗りつぶせない問題が起きていた．アンケートにおいても，塗りつぶしたい領域をすべて囲うのは面倒だというコメントを得た．

また UI については，ブラシサイズの変更バーが画面端にあることで使いにくい，左利きでは使いにくいといったコメントも得られた．また，9 人の実験参加者が操作に迷ったと答えた，情報を提示するなどの改善が必要だと考えられる．液晶タブレットは利き手によって使いやすい UI の配置が異なるため，操作パネルを自由に変更することができる柔軟な UI が必要であると考えられる．

(2) 作業経験に関する考察

経験者について，使いやすさの 5 段階評価は 3.86 となり，全体と比べ 0.14 ポイント減少した．また，自動で塗り分けした結果について満足かの設問については，4 人が満足と答えたが，3 人が満足ではないと答えた．普段からイラストを描いており，レイヤ分けを行っている経験者は，自動レイヤ分け機能に要求する精度が高いためこのような結果になったと考えられる．ボタンの位置やツールバーの位置は適切かについては，4 人が不満を抱えていることがわかった．これは普段使用しているソフトと比較を行っていることからこのような結果になったと考えられる．実際のコメントとして，ブラシの調節ウィンドウはペンに追従した方が使いやすい，左右に色の変更などの操作アイコンを配置して欲しいという意見を得た．

初心者について，使いやすさの 5 段階評価は 4.31 となり，全体と比べ 0.31 ポイント上昇した．また，自動で塗り分けした結果について満足かの設問については，14 人が満足と答えた．ボタンの位置やツールバーの位置は適切かについては，5 人が何かしらの不満を抱えていることがわかった．実際のコメントとして，液晶タブレットは左利きだと使いにくく，カーソルが見えないといった意見を得た．これは本システムの UI の問題に加えて，ハードウェアの液晶タブレットについての意見も含まれていると考えられる．

6.4 成果物に対する考察

ある実験協力者に本システムと既存のグラフィックソフトを用いてレイヤ分けを行った際の成果物を図 20 に示す．成果物に対して評価を行った結果，本システム，既存のグラフィックソフトともに許容範囲ではあるが間違いが存在することを確認した．確認できた間違いはパーツの間違い，塗り忘れ，はみだしである．それぞれについて発生した原因について考察する．

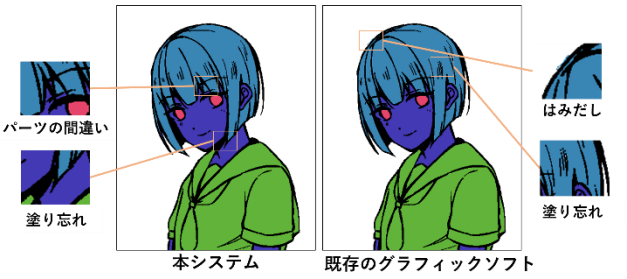

図 20　成果物に存在する誤り

(1) パーツの間違い

パーツの間違いとは，例えば前髪部分は肌パーツであるにもかかわらず，髪パーツを間違えて塗っていることである．これは既存のグラフィックソフト，本システムを使用した時の両方に見られたが，本システムを使用した際に頻出していたことを確認した．これは実験協力者が本システムの自動レイヤ分けの結果から，間違っている領域を発見できずに保存ボタンを押してしまったことが原因であると考えられる．また，パーツの間違いは前髪と眉の境界パーツで頻出していた．そのことから，髪と肌の色が似ている点や，線画において眉が前髪にかかっており，髪と肌の境界が認識しにくい点が原因であると考えられる．

(2) 塗り忘れ

塗り忘れとは，髪の毛の先端や小さな領域が塗り切れていないものを指す．これは本システムと既存のグラフィックソフトどちらにも確認できた．これは実験協力者が細かな領域を認識できなかった際に発生すると考えられる．認識できない理由としては，線画付近の色の間違いに気づかないことが挙げられる．解決方法として，レイヤ分けに使用する色を認識しやすい色にすることで，塗り忘れが少なくなると考えられる．

(3) はみだし

はみだしとは，領域から色がわずかにはみ出したものを

指す. これは既存のグラフィックソフトのみに確認できた．原因としては，ブラシツールを用いて修正を行った際に，線画から色がはみ出してしまい，はみだした色を削除せずに保存したことが原因であると考えられる．また，はみだしについては本システムにおいては発生しなかった．本システムはブラシツールを使用せずにレイヤ分けを行うことができることから,そのような結果になったと考えられる．

7. おわりに

本論文では，デジタルイラスト制作におけるレイヤ分け作業を対象として，pix2pix の出力結果に対して後処理を行うことで，レイヤ分け処理を自動化する方式について提案した．提案方式について精度評価を行った結果，Mean Accuracy で平均 84.8%の精度が得られた．次に，提案方式を用いた自動レイヤ分け処理において誤りが発生した場合，発生した誤りを手動で修正する UI を持った自動レイヤ分けシステム:Smart Layer Splitter を開発し，そのシステム評価を行った．

その結果，既存のグラフィックソフトと比較して，作業時間を 39.8%短縮できるとともに，操作回数を 68.6%削減できることが確認でき，システムの有効性について確認することができた．また，ユーザビリティの評価の結果，本システムを使用した実験参加者全員が最後までレイヤ分けを行うことができ，使いやすさについても 5 段階評価で 4 という良好な結果が得られた．

今後の課題としては，機能面については，アルゴリズムの見直しによる自動レイヤ分け処理に対する更なる精度の向上，今回対応できなかった色のあふれに対応した修正 UI の開発，ユーザにとって視認しやすいレイヤ分けに使用する色の検討があげられる．また，今回，システム評価にて実施したアンケートでは，システムの操作に迷った実験参加者が 9 名いたことから，システム上に操作の手順をわかりやすく提示する UI の修正についても検討していく予定である．

謝辞 本研究を推進するにあたり，有益な助言を頂いた東京電機大学 システムデザイン工学部 倉持卓司教授，阿部清彦准教授，及び開発したシステムの評価実験に協力頂いた東京電機大学 システムデザイン工学部の学生各位に感謝する．

参考文献

[1] P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros: “Image-to-Image Translation with Conditional Adversarial Networks”, Proc. of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR2017), pp.5967-5976 (2017).
[2] Preferred Networks: “PaintsChainer”, https://paintsChainer.preferred.tech/.
[3] LvMin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji and ChunPing Liu: “Two-stage Sketch Colorization”, ACM Transactions on Graphics, Vol.37, Article No. 6 (2018).
[4] C. Furusawa, K. Hiroshiba, K. Ogaki, Y. Odagiri: “Comicolorization: Semi-Automatic Manga Colorization”, Proc. of ACM SIGGRAPH Asia 2017, Technical Briefs (2017).
[5] Paulina Hensman and Kiyoharu Aizawa: “cGAN-based Manga Colorization Using a Single Training Image”, 14th IAPR International Conference on Document Analysis and Recognition, Workshop MANPU2017, pp.72-77 (2017).
[6] O. Ronneberger, P. Fischer, T. Brox: “U-Net: Convolutional Networks for Biomedical Image Segmentation”, Proc. of 18th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI2015), Springer, LNCS, Vol.9351, pp. 234–241 (2015).
[7] S. Ramassamy, H. Kubo, T Funatomi, D Ishii, A. Maejima, S. Nakamura, Y. Mukaigawa: “Pre- and Post-Processes for Automatic Colorization using a Fully Convolutional Network”, Proc. of ACM SIGGRAPH Asia 2018, Posters (2018).
[8] E. Simo-Serra, S. Iizuka, H. Ishikawa: “Real-Time Data-Driven Interactive Rough Sketch Inking”, ACM Transactions on Graphics, Vol.37, Issue 4, Article No. 98 (2018).
[9] 渡邉優，阿部清彦，阿倍博信: “pix2pix を用いたデジタルイラスト制作におけるレイヤ分け作業の自動化”，情報処理学会 第 81 回全国大会 4Q-02 (2019).
[10] 渡邉優，阿倍博信: “デジタルイラスト制作の色塗り工程における自動レイヤ分け方式”, Visual Computing 2019 P40 (2019).
[11] Adobe Systems: “The Photoshop File Format”, https://www.adobe.com/devnet-apps/photoshop/fileformatashtml/#50577409_72092/.
[12] pixiv: “pixiv”, http://www.pixiv.net/.
[13] Preferred Networks : “Chainer”, https://chainer.org/.
[14] Preferred Networks : “chainer-pix2pix”, https://github.com/pfnet-research/chainer-pix2pix/.
[15] P. S. P. Wang, Y. Y. Zhang.: “A Fast and Flexible Thinning Algorithm”, IEEE Transactions on Computers, Vol.38, No.5, pp.741-745 (1989).
[16] E. Shelhamer, J. Long, T. Darrell: “Fully Convolutional Networks for Semantic Segmentation”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 39, No. 4, pp.640-651 (2017).
[17] 渡邉優，阿倍博信: “pix2pix を用いたデジタルイラスト制作における自動レイヤ分けシステム”，情報処理学会 デジタルコンテンツクリエーション研究会 第 24 回研究会, Vol.2020-DCC-24, No.27, pp.1-8 (2020).
[18] 渡邉優，阿倍博信: “Smart Layer Splitter：デジタルイラスト制作の色塗り工程における自動レイヤ分けシステム”，映像情報メディア学会 映像表現・芸術科学フォーラム 2020, Vol.44, No.10, AIT2020-146, pp.317-320 (2020).