Vol.2020-GN-109 No.27
Vol.2020-CDS-27 No.27
Vol.2020-DCC-24 No.27
2020/1/24

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

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を提案する．提案方式を適用した自動レイヤ分けシステムを開発し，有効性について評価した結果，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 68.8%削減できることが確認できた．

キーワード：デジタルイラスト，レイヤ分け，pix2pix，創作活動支援，AI 応用

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

YU WATANABE[†1]　HIRONOBU ABE[†2]

Abstract: In the color painting process, which is one process of digital illustration production, layer splitting line drawings into parts such as hair and skin is required. However, there was a problem that it took time and effort when using the existing graphic software fill tool. In this paper, we propose a method for automating layer splitting using pix2pix, a conditional GAN method. We developed an automatic layer splitting system using the proposed method and evaluated its effectiveness. As a result, it was confirmed that the working time was reduced by 39.8% and the number of operations could be reduced by 68.8% compared with the existing graphic software.

Keywords: digital illustration, layer splitting, pix2pix, supporting creative activities, AI application

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ず，対応として人手で線画の隙間をふさぎ修正した後，再度塗りつぶしツールを使用するか，塗りつぶした後に消しゴムツールではみ出した部分を消去する必要があるため，修正作業に手間がかかってしまうという問題があった．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ため，この作業を 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，パーツ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 1 種類であるが，style2paints はライティング画像，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．

Comicolorization[4]は漫画の自動彩色システムである．モ

1 東京電機大学
Tokyo Denki University

Vol.2020-GN-109 No.27
Vol.2020-CDS-27 No.27
Vol.2020-DCC-24 No.27
2020/1/24

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り，PaintsChainer や style2paints とは異なり GAN は使用していない．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を塗る．

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[5]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[6]．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，特長技術として，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の補正技術を追加することで，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[7]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[6]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，機械学習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ることで，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 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[8], [9]. 図 1 において，まず，ユーザが入力した線画を，pix2pix で学習したモデルを用いて，髪，肌，目，服の 4 つのパーツに分ける形で色を塗る．pix2pix の実行結果は，色のはみだしなど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難しい．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．

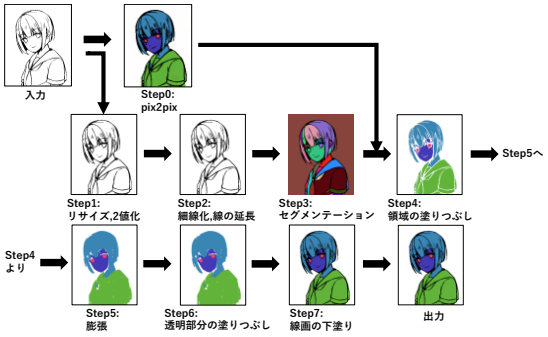

図 1　pix2pix を用いた自動レイヤ分け方式の概要
Figure 1　Overview of an automatic layer splitting method using pix2pix.

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．

- キャラクタの仕様

  上半身のみが映っているイラストを対象とし，2 頭身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も制限しない．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や目の中を塗りつぶしていても良い．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，後ろ向きの画像は対象外とする．

- 線の仕様

  線のスタイルは指定しない．髪などの一部の領域をベタで塗りつぶしている画像を含んでいても良い．漫画のトーンは含まない．ラフのような跡切れが大きい線や髪の毛において毛束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，線をふさぐことができな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レイヤ分け画像とし，レイヤ分け画像のフォーマットはレイヤ状態を保存でき，一般的なグラフィックソフトで扱うことが可能な PSD (PhotoShop Document)[10]形式とする．レイヤは上から線画，目，服，髪，肌，背景の順とする(図 2).

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

Vol.2020-GN-109 No.27
Vol.2020-CDS-27 No.27
Vol.2020-DCC-24 No.27
2020/1/24

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．

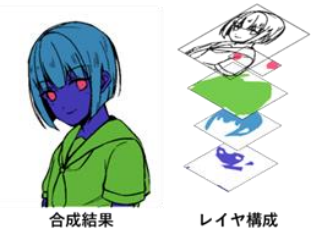

図 2　出力画像の概要

Figure 2　Overview of output image.

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[6]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．

(2) 学習データの概要

まず，イラストコミュニケーションサービス pixiv[11]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，透明部分を白で塗りつぶし，RGB の 3 チャンネルとした．収集した線画を，手動で肌，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おり，画像内には，背景，線，4 パーツの色のみが含まれている．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，作成したデータは合計で 13,800 枚となった．最後に，作成したデータを 9:1 に分け，9 割(12,420 枚)を学習用データ，残りの 1 割(1,380 枚)をテストデータとした．作成した学習用データの例について図 3 に示す．

図 3　学習用データの例

Figure 3　Example of training data.

(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[12]を選択し，chainer-pix2pix[13]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，入力画像は 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 4 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．また，80,000 iterator 前後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．

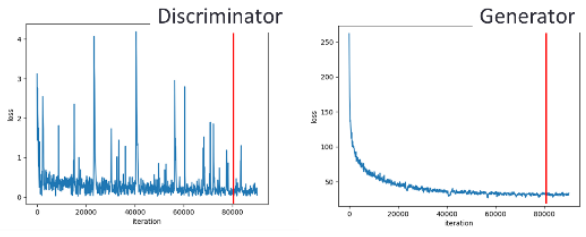

図 4　損失関数の出力の推移

Figure 4　Transition of loss function output.

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 5 に示す．図 5 を見てもわかるとおり，pix2pix の出力結果は同一の領域内が単色ではないことがわかるが，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．そのため，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．

Vol.2020-GN-109 No.27
Vol.2020-CDS-27 No.27
Vol.2020-DCC-24 No.27
2020/1/24

図 5　pix2pix の出力

Figure 5　Output of pix2pix.

以下，図 1 で示した Step ごとに，後処理の処理内容について説明する. 今回, 後処理は 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．具体的には，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. 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

Step2 では，2 値化した入力線画を Nagendraprasad-Wang-Gupta のアルゴリズム[14]を用いて細線化を行う．細線化画像から端点を検出し，隣接する線を 5px 辿り，コピーを行う．コピーした線を端点に貼り付けることで線の方向を延長し，線の跡切れを軽減する．これによって 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．

(3)　Step3:セグメンテーション

Step2 の結果と Step1 の線画を結合し，その結果を Python の画像処理ライブラリである pillow の fill 関数を用いて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．また，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 Python の pillow を用いて 3px の膨張を行う．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，その領域を Python の pillow を用いて塗りつぶす．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

線画の下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[15]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

Table 1　Result of accuracy evaluation of automatic layer splitting method.

	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，まずは，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分析

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た．また，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．

次に，自動レイヤ分け処理に失敗した箇所について原因

Vol.2020-GN-109 No.27
Vol.2020-CDS-27 No.27
Vol.2020-DCC-24 No.27
2020/1/24

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 6 に示す．

(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 6 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．

図 6　自動レイヤ分け結果の誤り

Figure 6　Automatic layer splitting result error.

(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の開発について述べる．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部分については，手動で修正する方針とする．また，今回，システムにて対象とする自動レイヤ分け処理の誤りは，領域の色の修正のみで対応が可能なパーツの間違いとする．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 のスペックを表 2 に示す．

表 2　PC のスペック

Table 2　Spec of PC.

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 7 に示す．

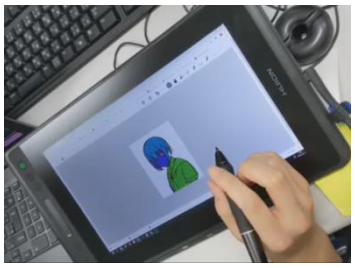

図 7　システムの外観

Figure 7　Appearance of the system.

次に，開発したシステムのソフトウェア構成について図 8 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 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．

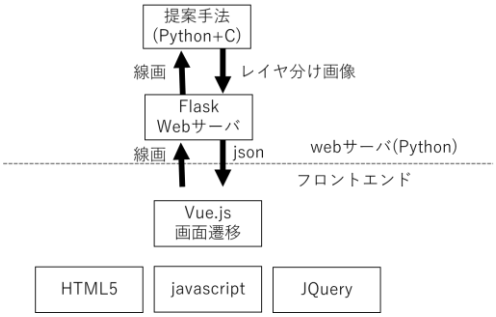

図 8　ソフトウェア構成

Figure 8　Software configuration.

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 9 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 9①)．次に，サーバで自動レイヤ分け処理を実行する(図 9②)．最後に，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存在する場合は，ユーザがブラウザ上で修正を行う(図 9③)．

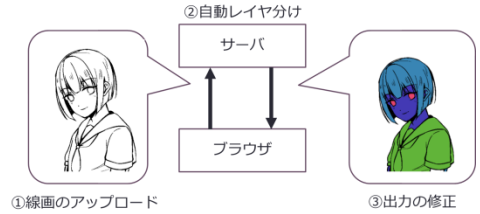

図 9　レイヤ分け処理の流れ

Figure 9　Flow of layer splitting process.

4.4 UI の概要

本システムの UI の概要を図 10 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．

Vol.2020-GN-109 No.27
Vol.2020-CDS-27 No.27
Vol.2020-DCC-24 No.27
2020/1/24

図 8　UI の概要

Figure 8　Overview of UI.

(1)　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．

(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)　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．

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

塗りつぶしツールの使用方法を図 11 に示す. 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．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 9　塗りつぶしツールの使用方法

Figure 9　Using the fill tool.

(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

選択ペンツールの使用方法を図 12 に示す．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，内部を塗りつぶすことができる．

図 10　ペン選択ツールの使用方法

Figure 10　Using the selection pen tool.

(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．

評価実験での課題を図 13 に示す．評価実験で使用した線画は, 図 13 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．

Vol.2020-GN-109 No.27
Vol.2020-CDS-27 No.27
Vol.2020-DCC-24 No.27
2020/1/24

図 11　評価実験での課題

Figure 11　Challenges in evaluation experiments.

5.2 作業時間と操作回数の評価

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測定した作業時間と操作回数について図 14 と図 15 に示す.

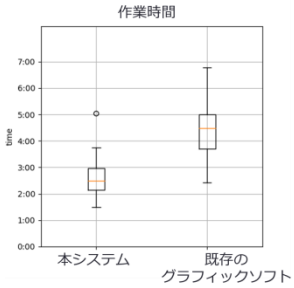

図 12　作業時間の評価結果

Figure 12　Results of the working time evaluation.

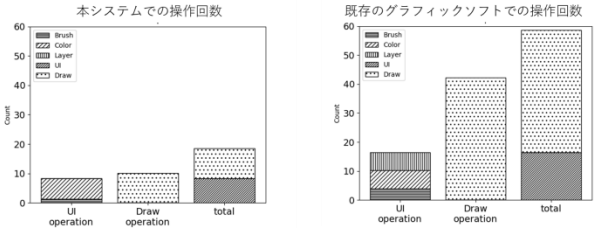

図 13　操作回数の評価結果

Figure 13　Results of the number of operations evaluation.

5.3 ユーザビリティ評価

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.

表 3　ユーザビリティの評価結果

Table 3　Results of usability evaluation.

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)

6. 考察

6.1 作業時間の評価に関する考察

図 14 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, 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.

6.2 操作回数の評価に関する考察

図 15 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，68.6%と大幅に削減することができた．各項目の内訳であるが，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%削減できることを確認した．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.

6.3 ユーザビリティの評価に関する考察

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.

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，まず，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を開発し，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 68.8%削減

Vol.2020-GN-109 No.27
Vol.2020-CDS-27 No.27
Vol.2020-DCC-24 No.27
2020/1/24

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，及び，今回対応できなかった色のあふれに対応した修正 UI の開発などがあげられる．

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]PreferredNetworks:”PaintsChainer”,https://paintsChainer.preferred.tech/.
[3]lllyasviel: ”style2paints”, https://github.com/lllyasviel/style2paints/.
[4]C. Furusawa, K. Hiroshiba, K. Ogaki, Y. Odagiri:” Comicolorization: Semi-Automatic Manga Colorization”, Proc. of ACM SIGGRAPH Asia 2017, Technical Briefs, (2017).
[5]O. Ronneberger, P. Fischer, T. Brox: “U-Net: Convolutional Networks for Biomedical Image Segmentation”, Prof of 18th International Conference on Medical Image Computing and Computer- Assisted Intervention (MICCAI2015), Springer, LNCS, Vol.9351, pp. 234–241, (2015).
[6]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).
[7]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).
[8]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).
[9]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).
[10]Adobe Systems: “The Photoshop File Format”, https://www.adobe.com/devnet-apps/photoshop/fileformatashtml/#50577409_72092/.
[11]pixiv: “pixiv”, http://www.pixiv.net/.
[12]Preferred Networks : “Chainer”, https://chainer.org/.
[13]Preferred Networks : “chainer-pix2pix” https://github.com/pfnet-research/chainer-pix2pix/.
[14]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).
[15]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).