色空間の変形による色彩調整方式

手塚忠則† 池田淳† 平島毅†
井上由紀子† 志水郁二†

従来から画像中の特定色のみを色彩調整する方式として,「画像上での領域選択」や,「色空間での調整領域（色域）の選択」を行う色彩調整方式が知られている．このような方式では，領域の選択に時間を要したり，疑似輪郭と呼ばれる画像中の不自然な色の変化が発生するという問題を有している．そこで，本研究では，わずか数点の色の指定で，疑似輪郭の発生を抑制しながら任意の色彩調整を可能とする色空間の変形による色彩調整方式（Synth4Color␣）を開発した．本論文では，Synth4Color␣のアルゴリズムと，そのソフトウェア実装について説明し，従来方式との違いを明確にするとともに，実装したソフトウェアの評価結果について述べる．

A Color Correcton Method by Utilizing the Color Space Transformation

TADANORI TEZUKA,† JUN IKEDA,† TUYOSHI HIRASHIMA,†
YUKIKO INOUE† and YUJI SHIMIZU†

This paper describes our color correction method, we called "Synth4Color␣", based on the color space transformation. Synth4Color␣ is the selective color correction method for digital image data, which can achive the color correction without selecting the image region or color space region. It only need to select a few points which designate the translation from a color to another color. In this paper, we describe Synth4Color␣ algorithm, and the software library "Synth4Color␣ Engine", and comparison with other methods, and result of speed evaluation of this software.

1. はじめに

近年の静止画像や動画像を利用した映像メディアの急速なデジタル化にともない，写真やグラフィックデザインの分野では広くデジタル画像編集が行われるようになっている．また，デジタルカメラやスキャナなどの入力機器，カラーインクジェットプリンタやビデオプリンタといった出力機器，そして編集機器であるパソコンの急速な低価格化も手伝い，デジタル画像処理を行う環境は，手軽に手に入れられるものとなった．

このように，デジタルで画像処理を行う環境は手軽に揃えられるようになった．しかし，画像のエフェクトや色彩の調整といった画像編集の操作の面でいえば，まだまだ経験や勘に頼る部分が多い．たとえば，画像の全体の色調を調整するという操作には,「明るさ・コントラスト調整」,「色相・彩度の調整」,「カラーバランス調整」，そして「レベル調整」など，非常に多くの調整方法が存在する．実際に意図した調整を行うためには，これらの調整方法をうまく組み合わせて利用する必要があり，どう組み合わせればよいかの判断は，経験や勘によるところが多い．

また，画像中の特定の色だけ色調を変更したい場合には，特定の色が含まれる画像中の領域を選択し，その後，選択した領域に対して調整を行ったり（以下，画像領域選択と示す），特定の色が含まれる色域の色空間上での領域を選択し調整する（以下，色域選択）といった複雑な操作をユーザに要求する．これらの作業は，デジタル画像処理に熟練したユーザであっても時間を要する作業である．

そこで，我々は，直観的な操作で手軽に利用できる色彩調整方法を目標とし，画像領域選択のような面倒な操作を必要とせず，かつ，複数色を同時に変更することを可能とする色彩調整技術 Synth4Color␣（シンセフォーカラースペース）を開発した．Synth4Color␣は，色空間の変形を用いた色彩調整方式で,「この色をある色に変えたい」という指定だけで色空間全体に対する写像関数を決定し，この写像関数を用いて調整後

† 松下電器産業株式会社九州マルチメディアシステム研究所
Kyushu Multimedia Systems Research Lab, Matsushita Electric Industrial Co., Ltd.

の色を決定する方法であり，わずか数点の指定によりユーザの意図する色調に調整することができる[1)]．

本論文では，色彩調整アルゴリズムについて詳しく説明するとともに，従来の色彩調整方式との違い，および，アルゴリズムに基づいて作成したソフトウェアの速度評価の結果について述べる．

2. 色彩調整アルゴリズム

Synth4Color␣は，変えたい色（以下，元の色と示す）を，何色にするのか（以下，変更色と示す）という指示だけで画像全体の色を調整する方式である．この1つの元の色とそれに対応する変更色の組を「調整点」と呼ぶ．Synth4Color␣アルゴリズムの特徴の1つは，この調整点が複数個指定できることである．調整点を複数個持てることで，複数の色を同時に違う色に調整することが可能であり，また，元の色と変更色が同じである調整点を設定することで特定の色を変更しないことも可能である．

2.1 基本アルゴリズム

Synth4Color␣では，色空間のうち3次元空間で表現される色空間を変換対象としている．このような空間としては，RGB 空間，CIE-XYZ 空間[2)]，CIE-LAB 空間[2)]，YCbCr 空間などがある．なお，以下では，3次元色空間上の任意の点（色）を (X,Y,Z) の形で表すこととし，調整点 i における元の色 S_i を (X_{si},Y_{si},Z_{si})，調整色 D_i を (X_{di},Y_{di},Z_{di}) と表すこととする☆．また，3次元空間の各軸については，それぞれ $0 \leq X,Y,Z \leq 1.0$ に正規化している．

Synth4Color␣アルゴリズムの基本的な考え方は，各調整点における移動量と呼ぶベクトル量を，各調整点との距離に応じて重ね合わせていくということである．ここで，調整点 i における移動量を $Q_i(X_{qi},Y_{qi},Z_{qi})$，色空間上の任意の色を $X(X_x,Y_x,Z_x)$，任意の色Xを色彩調整した後の色を $Y(X_y,Y_y,Z_y)$ と表すと，前述した足し合わせは以下の式で表すことができる．

$$Y = \sum_{i=0}^{N} l(X,S_i)\cdot Q_i + X \qquad (1)$$

式(1)は，調整後の色 Y が，調整前の色 X にすべての指定点との距離に応じた影響量を重ね合わせたものであることを示している．また，$l(X,S_i)$ は色空間における点 X と元の色 S_i との距離に応じて減少する関数であり，ここでは，これを距離関数と呼ぶ．

☆ ここでの (X,Y,Z) は，任意の3次元色空間を表すもので，CIE-XYZ 空間には対応しない．

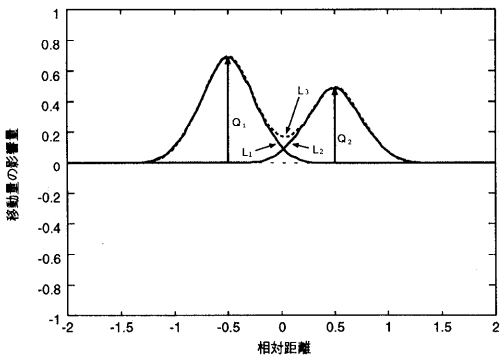


図1 調整点が2つの場合の影響量
Fig. 1 A effect of the case of two designated points.

距離関数 l は，たとえば式(2)のような関数である．式(2)は，距離関数の一例であり，本方式では，距離関数を適当に設定することにより，色彩調整の結果を目的に応じて変更することが可能である．

$$l(X,Y) = \begin{cases} 0.5 + 0.5\cos(3.14\cdot e(X,Y)), \\ \qquad e(X,Y) < 1.0 \text{ のとき} \\ 0, \qquad e(X,Y) \geq 1.0 \text{ のとき} \end{cases} \qquad (2)$$

※ただし，$e(X,Y) = \sqrt{(X_y-X_x)^2+(Y_y-Y_x)^2+(Z_y-Z_x)^2}$

次に，この重ね合わせを分かりやすくするために，横軸に距離を，縦軸に影響量をとる2次元グラフを用いて説明する．

図1は，調整点が2つの場合の影響量を示したものである．各調整点での移動量は Q_1，Q_2 であり，図では，2つの調整点からの影響は，それぞれの点からの距離に応じて L_1，L_2 の曲線で示されるように減少している．なお，この L_1 と L_2 の形状は，前述した距離関数に対応したものである．式(1)から分かるように，色空間の各点での合成影響量は，各調整点からの影響量のグラフ L_1 と L_2 を合成したものとなる（L_3）．

Synth4Color␣では，このように各調整点で最大となる調整点ごとの影響量を合成し，色空間上の任意の点Xが色彩調整後に何色になるのかを決定する．

以下に，移動量 Q_i の求め方について説明する．まず，調整点 i における元の色を変更色（色彩調整後に得たい色）に変えるために必要な色空間上でのベクトル量を調整量 P_i とし，以下のように定義する．

$$\begin{aligned} P_i &= (X_{pi},Y_{pi},Z_{pi}) \\ &= D_i - S_i \\ &= (X_{di}-X_{si},Y_{di}-Y_{si},Z_{di}-Z_{si}) \end{aligned} \qquad (3)$$

各調整点 i での調整量 P_i が式(1)により演算した

移動量と一致するという条件から，式(4)を導くことができる．

$$D_i = P_i + X_i \tag{4}$$

次に，式(1)と式(4)を用いてさらに変形を行うと，以下の移動量 Q_i と調整量 P_i の条件式が求められる．

$$P_i = \sum_{j=0}^{N} l(S_i, S_j) \cdot Q_j \tag{5}$$

式(5)により，各調整点 i での調整量 P_i と式(1)の演算結果が一致する移動量 Q_i を求めることができる（式(6)）．

なお，式(6)では，P_0 から $P_n - 1$ までの調整量をベクトル P，Q_0 から $Q_n - 1$ までをベクトル Q と表記している．また，$l(S_i, S_j)$ を l_{ij} と表記した．

$$Q = \begin{pmatrix} l_{11} & \cdots & l_{1n} \\ \vdots & \cdots & \vdots \\ l_{n1} & \cdots & l_{nn} \end{pmatrix}^{-1} \times P \tag{6}$$

Synth4Color␣アルゴリズムでは，この式(6)を用いて移動量を演算し，演算された移動量を用いて色空間上の任意の点が何色に変化するかを演算する．

ここで，以上のアルゴリズムを，3次元グラフを用いて説明を行う．ここで用いる3次元グラフは，X–Y平面が色空間上のある平面を表している．また，Z軸はこの平面上の各点の合成影響量を表している．なお，説明を簡単にするために，ここでは，すべての調整点がこのX–Y平面上にあるとし，この平面の色彩調整による変化だけを示した．

図2は，調整点が1つの場合の合成影響量を示したものである．なお，図2では，調整点は平面上の(0,0)にあり，調整量は1であるとした．調整点が1つの場合には，調整量 P_i ＝ 移動量 Q_i となり，調整点以外の点は，距離関数に応じた調整量の影響を受けることになる．ここでは，距離に応じて単調減少する距離関数を用いたので，図2は調整点を頂点とした単調減少のグラフとなっている．このグラフの形は，色空間上の特定領域のみの色を変化させる色域選択方式において，境界部分を目立たなくするために，境界をなめらかに変化させた場合と似た形状となる．

本アルゴリズムでは，式(1)が示すように，さらに他の調整点を指定することが可能である．

図3は，図2に加えて点(1,1)を新たに調整点として加えた場合の合成影響量のグラフである．なお，新たな調整点の調整量は -0.5 とした．図3では，2つの調整点での合成影響量は，それぞれ指定した調整量と同じ1と -0.5 となっている．これは，ユーザが調整点として指定した部分の色（元の色）が，ユーザが指定した色にきちんと調整されたことを意味している．

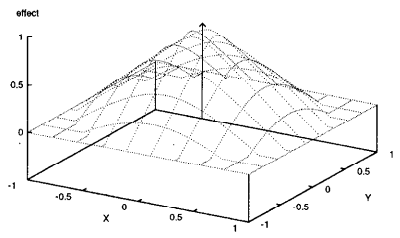


図2 調整点が1つの場合の影響量（3次元）
Fig. 2 A effect of the case of a designated point (3D).

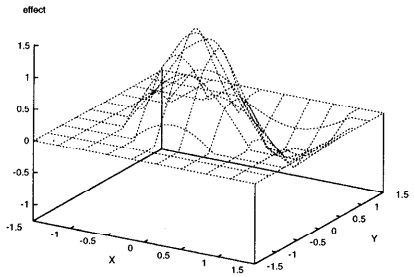


図3 調整点が2つの場合の影響量（3次元）
Fig. 3 A effect of the case of two designated points (3D).

また，図3から，2つの調整点以外の部分については，2つの調整点の間が滑らかにつながるように移動していることが分かる．このように，色空間上で滑らかに移動していれば，この変換により色彩調整された画像の色味も同じように滑らかに変化することになる．したがって，疑似輪郭のような極端な色の変化が調整後の画像に発生しにくいといえる．

2.2 距離関数について

本方式において距離関数の決定は非常に重要な部分である．先のアルゴリズムの説明では距離関数として式(2)を利用して説明を行った．しかし，この距離関数を用いた場合，利用者によって指定された調整点間の距離が極端に近い場合に問題が生じる．**図4**は，2つの調整点の距離が近い場合の例である．このグラフでは，0と0.1の2つの点に調整点を指定し，それぞれの0.5および -0.2 だけ色を変更したものである．これを見ると，この2つの点の左右の部分では正規化された色空間の領域を大きくはみ出している部分が存在することが分かる．このはみ出した部分を色空間の範囲内にクリッピングするとすれば，この部分が単色になり，全体としてべったりした画像になってしまう．このように調整点間の距離が近い場合には利用者の意図しない色彩の変化が発生してしまうため，この距離

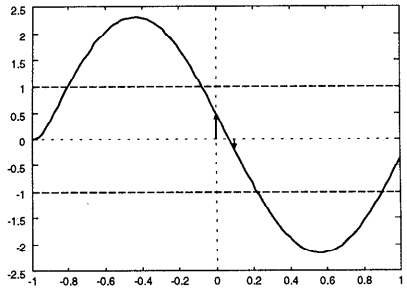


図4 式(2)を距離関数とした場合
Fig. 4 A case of $l(x,y) = 0.5 + 0.5\cos(\ldots)\ldots$.

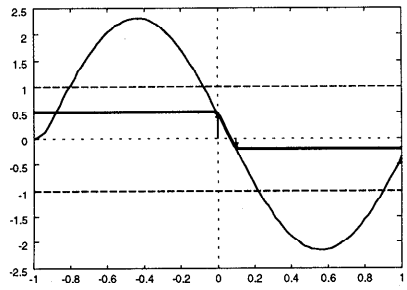


図5 式(7)を距離関数とした場合
Fig. 5 A case of $l(x,y) = 1 - x/10000\ldots$.

関数を利用するのは問題である．

そこで，この問題を解決するために，実装を行ったソフトウェアでは距離関数として式(7)を利用した．

$$l(X,Y)=\begin{cases}1-e(X,Y)/10000, \\ \quad e(X,Y)<10000 \text{ のとき} \\ 0, \quad e(X,Y)\geq 10000 \text{ のとき}\end{cases} \quad (7)$$

※ただし，$e(X,Y) = \sqrt{(X_y-X_x)^2+(Y_y-Y_x)^2+(Z_y-Z_x)^2}$

この関数は，非常に緩やかな単調減少関数であり，1つの調整のみが指定された場合には，色空間全体がほぼ均一に変化する．**図5**は，図4に距離関数として式(7)を利用した場合のグラフを追加したものである．これから分かるように，式(7)を利用した場合は，前述した調整点が近い場合に発生する問題を解消することができる．

また，1点だけ調整点を指定した場合には画像全体の色が変化するようになるので，まず変えたい色を指定し，その後変わってほしくない色を「押える」という直観的な操作が可能になる．

2.3 色空間について

ここでは，調整を行う色空間の違いが調整結果に与える影響について説明する．

色空間は大きく分けて，色相/濃さ/明るさの3つのパラメータで表現される円筒形または円錐形の色空間（以下，円筒空間とする）と，明るさと2つの色差で表現されるような直交座標系の色空間（以下，直交空間とする）の2つに分類することができる．

本方式は，どちらの色空間においても色彩調整を行うことが可能である．しかし，円筒空間の場合には色相方向に沿って回転させた場合の距離と，円筒空間上の直線距離のどちらを「距離」として利用するかにより，調整結果が大きく変化する．また，円筒空間の場合，色空間での移動は色相方向の回転と濃さ/明るさ方向への拡縮で表現されるため，回転や拡縮の中心軸となる無彩色を有彩色に変えることができない．このため，色空間上のすべての色を任意の色に調整可能としたい場合には，直交空間を利用する必要がある．

また，画像の色空間と調整を行う色空間の色域が異なる場合，調整結果を再び画像の色空間に変換するときのクリッピングにより色潰れが発生したりするという問題がある．たとえば，RGB空間で表現される画像をCIE-La*b*空間で調整処理を行い，再びRGB空間に戻したときにこの問題が発生する．これは，CIE-La*b*空間の方がRGB空間よりも広いためである．

利用者が操作するインタフェースは明るさや濃さといったパラメータで操作する方がRGBなどで操作するより使いやすく，調整する空間と利用者の操作する空間は近い方が意図した結果に近い結果を得ることができると考え，以下のソフトウェアでは色空間としてYUV空間を利用した．

3. ソフトウェア構成概要

3.1 ライブラリ概要

ここでは，Synth4Color␣アルゴリズムをソフトウェアライブラリとして実装したSynth4Color␣ Engineについて説明を行う．Synth4Color␣ Engineは，先に説明したアルゴリズムをアプリケーションプログラムに組込み可能なライブラリとして実現したものである．

なお，ライブラリでは，色彩調整処理を行う色空間をRGB空間から式(8)で変換できるYUV空間とした．

$$\begin{aligned} \mathrm{Y} &= 0.299\,\mathrm{R} + 0.587\,\mathrm{G} + 0.114\,\mathrm{B} \\ \mathrm{U} &= -0.1687\,\mathrm{R} - 0.3313\,\mathrm{G} + 0.5\,\mathrm{B} \\ \mathrm{V} &= 0.5\,\mathrm{R} - 0.4187\,\mathrm{G} - 0.0813\,\mathrm{B} \end{aligned} \quad (8)$$

YUV空間を用いた理由は，先に説明したとおりである．なお，式(8)はRGBの値の範囲を0.0～1.0とした場合の変換式である．

今回実装したライブラリは，その処理内容により大

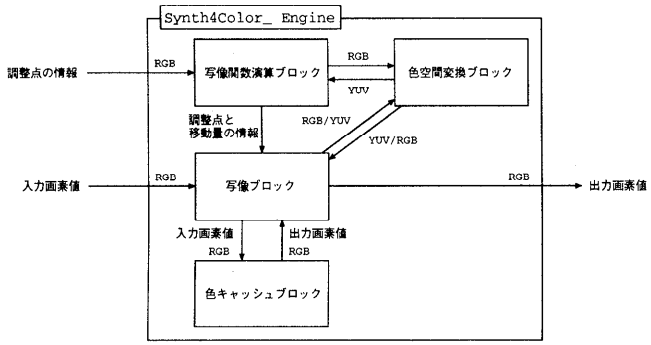


図6 ソフトウェア構成（Synth4Color Engine）
Fig. 6 A software block (Synth4Color Engine).

きく4つのブロックに分けることができる．**図6**は，実装したライブラリのソフトウェア構成を示すものである．図6から分かるように，Synth4Color␣ Engineが外部から入力する情報は，各調整点の情報と，色彩調整を行う前の画像の画素の値（入力画素値）だけであり，出力する情報は色彩調整後の画素値（出力画素値）である．各調整点の情報は，元の色と変更色が組になった情報であり，これがユーザが指定した調整点の数だけ入力される．なお，入出力に関しては，調整点の情報も含めてすべてRGB空間の値によって受け渡される．この，RGB空間の値の範囲は一般的な画像データとして利用されている，各8 bitで0～255の範囲の値である．

ここで，色空間変換ブロックは，画素値のRGB–YUV間の相互変換を行い，写像関数演算ブロックは，入力された調整点の情報から移動量 Q_i を演算している．また，写像ブロックは，画素値を入力し，対応する調整後の画素値を演算し出力する．色キャッシュブロックでは，調整結果を記録しておき，同じ色の画素値が入力された場合にはキャッシュに記録された出力画素値を出力することで，色彩調整に要する処理時間の短縮を図っている．

3.2 グラフィカルユーザインタフェース（GUI）の実装

ここまで，Synth4Color␣ Engineのソフトウェア構造について説明した．Synth4Color␣ Engineはソフトウェアライブラリであるため，それ単体ではアプリケーションソフトウェアのように実行することができない．

そこで，Synth4Color␣ Engineの評価のために，実行可能なソフトウェアを作成した．作成したソフトウェアは，**図7**に示すように市販フォトレタッチソフトウェアのAdobe Photoshop用のプラグインとして動作するもので，色彩調整を行うためのGUIを備える．

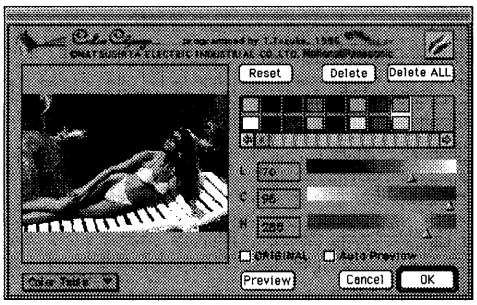


図7 プラグインの画面キャプチャ
Fig. 7 A screen capture of plug-in.

このソフトウェアでは，表示されたサンプル画像をマウスでクリックし（調整点の選択），調整後の色を明るさ，色の濃さ，色相の3つのスライダーを操作して決定するだけである（調整色の決定）．以降の部分では，このソフトウェアを利用して評価を行っている．なお，実行環境は以下のとおりである．

実験環境（UMAX Pulsar 225：マッキントッシュ互換機）

- プロセッサ：PowerPC 604e（225 Mhz）
- メモリ：288 Mバイト
- ディスク容量：2 G+1 Gバイト
- OS：MacOS 7.5.5
- Photoshopバージョン：4.0J

4. 評　　価

4.1 従来技術との比較

部分的な色彩調整を行う従来からの代表的な手法としては，(1) 画像の領域をマウス操作などでユーザが指定する方式（以下，画像領域選択方式）と，(2) 色空間上で領域を選択し，選択された領域のみの色彩の調整を行う方式（以下，色域選択方式）がある．ここでは，この2つの方式との本方式の比較を行った結果

について述べる．

4.1.1 画像領域選択方式

画像領域選択方式を説明するために，まず，代表的な画像処理ソフトにおける領域の選択手順について簡単に示す．画像処理ソフトの中には，画像中の特定の場所をマウスでクリックすると自動的に似た色の部分を選択する機能を持つものもあるが，このような機能による選択だけで意図した部分の色を完全に選択できることは少ない．このため，画像領域を選択するためには，マウスなどのポインティングデバイスを利用して画像中の特定領域を囲むという方法が行われることが多い[3]．

図8 は，マウスを用いて手作業で紫色の花の部分だけを選択した例である．図8において，点線で囲まれた花の部分が選択された部分であり，灰色に塗りつぶされた部分が非選択部分である．このような複雑な領域の選択は，熟練したユーザであっても非常に時間のかかる作業が必要である．

一方，Synth4Color␣において，前述したGUIを用いて同じく紫色の花の部分だけの色を変化させる場合，指定した調整点の数は全部で9点であった．なお，9点には，色を変えたくない点，つまり元の色 = 変更色と指定した調整点も含まれている．**図9** は，元画像と調整画像の差分を画像にしたものである．図8と図9を比較すると，どちらの場合もほぼ同じ領域が選択されているが，Synth4Color␣では領域の境界がなめらかなグラデーションになっていることが分かる．画像領域選択方式の場合も，同じように領域の部分について選択を行うことができるが，この場合，選択範囲が複雑になり，ユーザの作業量の面から考えた場合現実的ではない．これに対して，Synth4Color␣では画像領域の選択の必要はなく，わずか数点の調整点の指定で複雑な形状をした部分の色彩の調整が可能である．

図10 は，図9の色彩調整をRGBの色空間で示したものである．図10において左側が元画像であり，右側が調整後の画像である．また，調整点のうち，元の色と変更色が異なる調整点（変更した色）を矢印で，元の色と調整点が同じ（変更したくない色）を四角で表現した．図10では，押えた色は調整後の画像でも押えられていること，調整点として指定されていない色は，調整点との関係に従って移動していることが分かる．

4.1.2 色域選択方式

先に説明したように，本方式は色空間の変形による色彩調整方式である．画像ではなく，色空間を対象としている点では，本方式は，画像領域選択方式よりも色域選択方式に近い方式だといえる．色域選択方式は，色空間の特定領域を切り出し，その部分に関してのみ色彩を変化させる方式である．**図11** は，HSV空間における色域の選択例である．図11では，各軸H，S，Vのある範囲の部分だけが選択されている．このように，色域選択方式では，色空間を簡単な幾何学図形で切り出すのが一般的である．なお，幾何学図形としては，指定された色を中心としたユニティ，台形，ガウス分布などが利用される[4]．

図12 および**図13** は，**図14** の画像の肌色の部分だけを濃くするように色域選択方式および我々の方式で調整した結果である．図12では，肌色の部分の色域を選択して調整を行ったが，肌の明るい部分と波しぶき色が近いために，波しぶきの部分の色も色域内に入ってしまい，結果として波の部分に疑似輪郭が発生している．

一方，図13では，明るい肌色の部分から波しぶきの色にかけて滑らかに色が変化するように調整されるため，疑似輪郭は発生しない．なお，肌色を濃くするために設定した調整点の数は，わずか4点で，このうち1点が肌色を濃くする設定で，残りの3点は変化させない（元の色 = 変更色）という設定であった．

4.1.3 方式比較のまとめ

領域選択方式と比較した場合，領域選択方式では領域を選択する手間が非常に大きく，これと比較した場合は本方式では短い時間で色彩調整を完了可能である．

一方，色域選択方式と比較した場合には，色域方式では領域を幾何学的な図形のパラメータとして渡すために，中心となる色，つまり選択する色が非常に重要になる．本方式と同様に画像から中心となる色を取ることもできるが，画像中から調整の中心に近い色を指定するのは難しい．しかし，我々の方式では，1) 変えたい色に加え「変えたくない」色が指定できること，2) 複数の色を同時に調整できることの2つにより，調整点として選択した色が適切な色と異なる場合でも調整結果が大きく異なることを防いでいる．もちろん，適切な色を選択した方が本方式で調整点の数を減らすことができるが，適切な調整点を指定しなくても調整点を増やすことで視覚的に近い調整結果を得ることができる．

さらに，作成したソフトのGUIでは，調整結果を確認しながら点を増やしたり減らしたりすることで，満足のいく結果になるまで調整を繰り返し行うことができる．これは，本方式が複数点を同時に調整可能であることで実現可能なインタフェースであり，これにより調整のための操作と手順が単純化できた．

4.2 速度評価

ここでは，Synth4Color␣ Engine を組み込んだ Adobe Photoshop のプラグインを利用して，速度評価を行った結果について述べる．なお，実験環境は 3.2 節で示したとおりである．

4.2.1 調整点の数による速度評価

本方式は，式 (1) から分かるように，調整点の数が増加すればそれだけ演算量が増加し処理速度に影響を与える方式である．そこで，画像サイズを 3072 × 2048 に固定し，調整点の数だけを増やした場合の処理時間の変化について測定を行い，調整点の増加によ

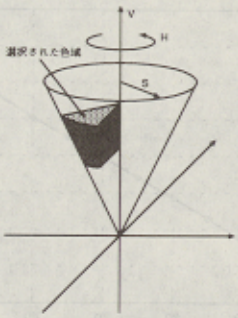


図 11 HSV 空間での色域選択例
Fig. 11 The example of the color region selection (HSV color space).

図 8 領域選択例
Fig. 8 The example of the region selection.

図 12 色域選択による色彩調整
Fig. 12 The color correction example by using a color region selection.

図 9 Synth4Color による変換例 (1)
Fig. 9 Synth4Color algorithm example (1).

図 13 Synth4Color による色彩調整
Fig. 13 The color correction example by using Synth4Color .

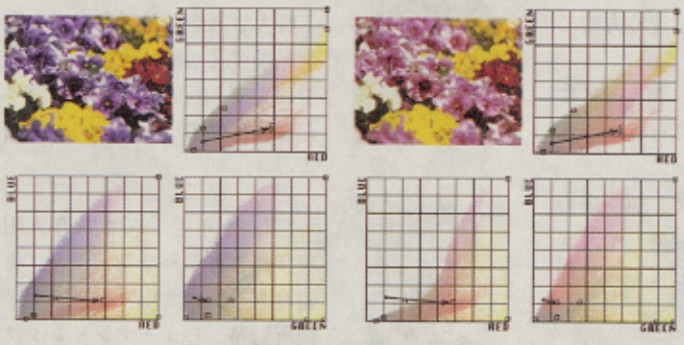


図 10 色空間の変形の例 (RGB 空間)
Fig. 10 A example of color space transformation (RGB view).

図 14 元画像
Fig. 14 The original image.

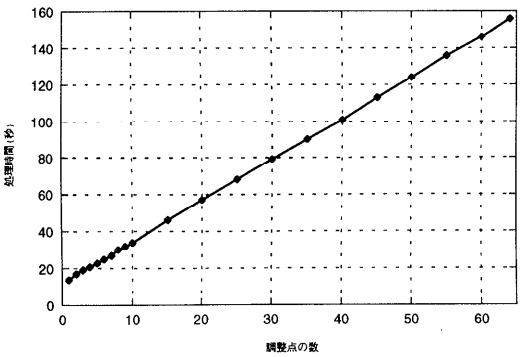


図15 調整点の数による処理速度の増加
Fig. 15 The processing speed vs. # of designated points.

り処理速度がどのように変化するのかを実験した．なお，画像データとしては，Kodak Photo CD Access Software & Photo Sampler に収録された画像の4枚目を用いた．

図15 は，調整点の数を変化させて処理時間を測定した結果のグラフである．調整点が1つの場合に14秒の処理時間がかかっているが，この中にはRGBのデータを4キロバイト単位でホストプログラムであるPhotoshopに要求し，受け取る処理のための処理オーバヘッドが含まれている．

図15から，調整点1つあたりの演算時間の増加は2秒程度であり，数点の調整点を指定した場合には，画像データの受渡しオーバヘッドに比べて処理時間は短いことが分かった．

5. まとめ

Synth4Color␣では，調整点の情報に基づいて色空間全体を変形させることにより，色を変化させたい部分と変化させたくない部分にかけて滑らかに色が変化するため，色域選択において発生していた疑似輪郭を抑制することができる．また，調整点を複数個同時に指定できるので，結果として複数色の色を違う方向に同時に調整することも可能である．

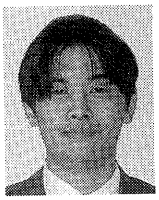
また，本色彩調整方式を組み込んだソフトウェアにおいては，画像に対する処理が十分高速に行えることを確認した．

今後は，このアルゴリズムの応用として，色空間の変換（たとえば，RGBからCMYへの変換）や，画像の特定色の切出し（マスク切出し）などについて検証を行う予定である．

参考文献

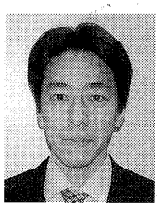
1) 手塚忠則，池田　淳，平島　毅，井上由紀子，志水郁二：簡易操作による色彩調整技術の開発，映像情報INDUSTRIAL 11月，Vol.28, pp.10–14 (1996).
2) 応用物理学会光学懇話会：色の性質と技術，朝倉書店 (1986).
3) アドビシステムズ：Adobe Photoshop 3.0Jユーザガイド，Adobe Systems Incorporated (1995).
4) 渡辺　立ほか：ワンタッチカラーコレクタの開発と運用，画像電子学会研究会予稿集，pp.17–20 (1993).

(平成9年5月26日受付)
(平成9年12月1日採録)



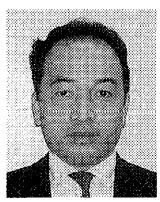
手塚　忠則（正会員）
昭和43年生．平成5年九州工業大学大学院情報科学専攻修士課程修了．同年松下電器産業（株）入社．ネットワークを用いた並列処理，および色彩調整技術の研究開発に従事．

池田　淳
昭和39年生．平成元年九州大学理学部物理学科卒業．同年，(株) 東芝に入社．平成3年松下電器産業（株）入社．現在，映像情報処理の研究に従事．映像情報メディア学会会員．



平島　毅
昭和42年生．平成4年九州大学大学院工学研究科電器工学専攻修士課程修了．同年松下電器産業（株）入社．現在，色彩情報処理に関する研究開発に従事．電子情報通信学会会員．

井上由紀子
昭和45年生．平成5年九州芸術工科大学画像設計学科卒業．同年松下電器産業（株）入社．色彩調整技術および，画像メディアに関する研究開発に従事．



志水　郁二
昭和32年生．昭和55年九州大学理学部物理学科卒業．現在，松下電器産業（株）九州マルチメディア研究所勤務．ヒューマンインタフェース等の研究開発に従事．電子情報通信学会会員．