カラリゼーションに関する基礎的研究

坂本　眞人[1)]・衣松　宏晃[2)]

Fundamental Study on Colorization

Makoto SAKAMOTO and Hiroaki KINUMATSU

Abstract

Colorization is a computerized process of adding color to a monochrome image or movie. The process was first used in 1970 to add color to monochrome films of the moon from the Apollo mission. However, colorization is an expensive and time-consuming process. Recently, various software of colorization are introduced on the Internet, but operation is difficult for a beginner. In this paper, we deal with the fundamental study on colorization, and show the program for colorization that even a beginner can use. We use C for implementation of program. In this process, we use the image which we appoint a color and the place that we want to paint on a monochrome image. We call it "pilot image". We need it besides monochrome image.

Keywords: Black-White Image, Colorization, HSV Color-Space, Pilot Image, RGB Color-Space

1. はじめに

1.1 研究背景

「色を着ける」ということは重要である。例として絵の着色を想像してほしい。色を塗る前の絵は、木がある、花がある、雲がある等のいわば「存在」の情報しか与えられていない。しかし、着色するとどうだろうか。空を赤く塗ると夕焼けになり、地面を緑で塗れば草原になる。つまり、色を塗ることで「存在」だけでなく「背景」をも表現することができる。これにより、白黒だけの絵ではわからない情報を読み取ることができる。

カラリゼーション(カライゼーション)という言葉がある。カラリゼーションとは1970年にWilson Markle氏が白黒画像やテレビの映像を着色するために開発したコンピュータ支援方法を表現するために導入された用語である[3)]。

カラリゼーションの使用用途の一つは、白黒写真のカラー化である。1800年代からカラー写真の開発が始まっていたが、日本の最初のカラー写真が開発されたのは1940年である[9)]。つまり、1940年以前の写真は白黒である。これらの写真をカラー化するためにカラリゼーションが使われている。カラー化することで、白黒画像では得られない情報を得ることができる。

カラリゼーションは膨大な処理時間とユーザの負担を必要とし、境界線を塗り分けた完璧な着色はできていなかった。現在、多くの技術者により処理時間とユーザの負担はかなり軽減されており、なおかつ完璧に近い着色ができつつある。だが、完成度の高いカラリゼーションソフトは高価であり、フリーソフトのものは少ない。

そこで、本研究はフリーソフトという前提で境界線を見極めた着色ができるカラリゼーションソフトのプログラム開発を行った。

1.2 着色方法

白黒画像の着色手法は大きく分けて2つあり、自動で着色するものと、ユーザが着色する色と場所を指定して着色する方法がある。本論文のプログラムは後者の手法を使う。

本論文のプログラムは白黒画像の他に、「種画像」を用意する必要がある。種画像とは、白黒画像に塗りたい色・場所を指定した画像である。種画像は、本論文のプログラムとは別の画像処理ソフトで白黒画像に塗りたい色・場所を描画する必要がある。本論文のプログラムは白黒画像と種画像の2つの画像を使って着色する。入力データとして白黒画像と種画像、出力データとして着色画像となっている。

2. 理論

2.1 プログラムの流れ

1) 情報システム工学科准教授
2) 情報システム工学科学部生

このプログラムは色空間であるRGBとHSVを使用する。

白黒画像のHSV情報を変換して着色していく。詳しい着色方法は「2.6. 着色」にある。

2.2 RGB

RGBは色の表現法の1つで、R（赤）、G（緑）、B（青）からなる加法混色の一種である[8]。R要素、G要素、B要素の3つの要素の値の組み合わせで色、その色の鮮やかさ・明るさが決まる(図1)。主色としてRGBが選択されているのは、人の目の色感知応答を最もよく近似するためである[1]。RGBはブラウン管や液晶ディスプレイ、デジタルカメラなどで画像再現に使われている。

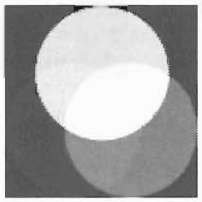

図1：RGB色空間.

RGBをデジタルデータとして扱う場合は、基本的にRGBそれぞれの色は0か255までの段階があり、8ビットデータとして扱う。つまり、一つの画素の色を表すためには赤と緑と青で24ビットのデータが必要になる。24ビットで扱える数値の範囲は0～16777215であり、これは16777216色もの色が使用できることを示す。

2.3 HSV

HSVは色の表現法の1つで(図2)、Hue(色相)、Saturation(彩度)、Value(明度)からなる[4]。

Hue（色相）とは、赤・黄・緑・青のような色の種類のようなもので、有彩色を分類することができる。赤・黄・緑・青・紫を順に並べて、赤と紫をつなげた輪、色相環で表現できる。

Saturation(彩度)とは、色の鮮やかさを示す尺度のことで、彩度が高いほど原色に近くなり、低いほど灰色さが目立ちくすんだ色になる。

Value（明度）とは、色の明るさを示す尺度のことで、白が最大の値をとり、黒が最小の値をとる。

H要素は色相環で表現しているので、一般的に値は色相環に沿ったディグリーの角度の値0～360となっている。

また、S要素・V要素はどちらも0～100%の範囲で表しているため、0～100までの値が一般的である。

HSVの要素の値はRGBの要素の値から算出できる。逆にHSVの要素の値からRGBの要素の値を算出できる[4,7]。

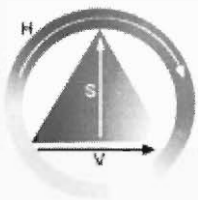

図2：HSV色空間.

2.4 画像の輪郭抽出

画像には輪郭が存在する。輪郭とは、物体の外縁をあらわす線、または画像を特徴づける線要素である。これを塗り分けるために輪郭抽出する必要がある。物体と物体、物体と背景の境目が輪郭なので、「画像の濃度や色に急な変化があるところ」が輪郭に見える。これにより、関数の変化分を取り出す微分演算が輪郭抽出に利用できる[2]。

本論文のプログラムでは、1次微分フィルタを使用する。1次微分フィルタにはPrewittフィルタとSobelフィルタがある。本プログラムでは、Prewittフィルタ、Sobelフィルタを使用して輪郭を抽出した画像を作成する。どちらのフィルタも似たような結果なのでどちらも使用できるようにした。

このプログラムでの閾値判定は、位置の異なる2つのピクセルにおいて、RGB要素の差の絶対値で判定する。絶対値が一定値以下ならば、同じ領域内にあると判定する。輪郭を抽出した画像と白黒画像の二つの画像で判定する。二つの判定でどちらも同じ領域と判定されたら、2つのピクセルは同じ領域だと判断される。

2.5 画像読み込み

本プログラムでは、プログラム実行時、画像を1ピクセル単位で読み込んでいく。便宜的にするため左から右へ、上から下の順で構造体にRGB情報を格納する。HSV情報も同様にして格納する。従って、本プログラムでは画像読み込みは左から右へ、上から下の順で読み込むことになる(図3)。

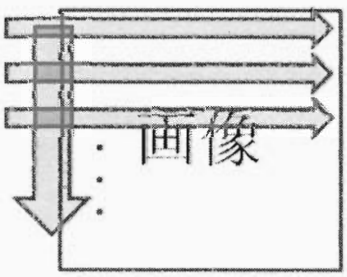

図3：画像の読み込み.

2.6. 着色

本プログラムはH変換、S増減、V増減の順で着色を行う。

2.6.1 H変換

H変換では、まず「描画された色探索」で種画像に描画された色を探索する。描画された色がなければ「着色された色探索」を行う。探索は白黒画像全てのピクセルに対して行う。

描画された色探索の説明をする。白黒画像と種画像、2つの画像のRGB情報を比較し、種画像に描画された色を探していく。この作業は白黒画像全てのピクセルに対して行う。探索方法は、白黒画像内にある対象ピクセルを中心とした正方形状の探索範囲で近くの描画された色を探す。描画された色があれば、描画された色のH値を白黒画像の対象ピクセルのH値に割り当てる。なければ探索範囲を拡げ探索していく(図4)。

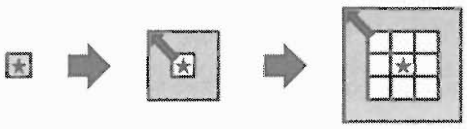

図 4：描画された色探索.

続いて着色された色探索の説明をする。この作業は歳代まで拡げた正方形内に描画された色がみつからない場合の救済措置の役割をする。この作業は2つの画像のHSV情報を比較し、近くの既に着色されたピクセルを探す。描画された色探索と同様のピクセル・同様の正方形で着色された色を探す。処理時間を短縮するために対象ピクセルの左方と上方だけを探索する(図 5)。着色された色があれば着色された色のH値を白黒画像の対象ピクセルのH値に割り当てる。なお、この時に閾値判定を行う。

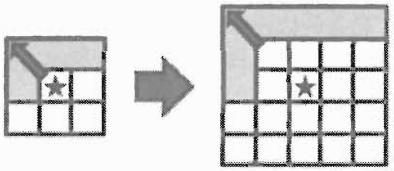

図 5：着色された色探索.

H要素には黒・灰・白の概念がないため、それに応じた処理を行う必要がある。それはH変換する際の描画された、または着色された色のS値とV値で黒・灰・白を判断する。その際、黒・灰・白と判断された場合、そのピクセルのS値を0にする。これは「2.6.2.S増減」の処理のためである。さらに、黒・灰のときV値を下げる。

種画像に描画された色を再現できるように、描画された色のS値によって対象ピクセルのS値を増加させる。増加させた値によって色の薄さを判断させる。描画された色のV値が低いと対象ピクセルのV値を下げる。

2.6.2 S増減

白黒画像はS値（彩度）が低いので、全てのピクセルにS値を増加させる必要がある。本プログラムでは、ピクセルのS値が0でない場合、定数を与える。ただし、描画された色を表現するために対象ピクセルのS値によって与える値は変化する。

もし、そのピクセルが白に近い、つまりS値が低くV値が高い場合、白さに応じてS値を下げるようにしている。

2.6.3 V増減

「2.6.2.S増減」と同様に白黒画像のV値は低い。従って、全てのピクセルに一定値を与える必要がある。本プログラムでは全てのピクセルに定数を与えている。

3. 実装

3.1 開発環境

以下の環境でプログラムを作成した。

プロセッサ：Intel(R) Core(TM) i7 2.93GHz
メモリ：4GB
OS：Windows Vista
プログラム言語：C
環境：Borland C++ Compiler
画像：BMP画像(24bitカラー)

3.2 実行手順

実行手順を以下に示す。

① 白黒画像を用意する。ただしBMP画像（24bitカラーでなければならない。
② ペイント等の画像処理ソフトで塗りたい色・場所を指定して種画像を作成する（図6）。

図 6：実行例 1[6].

③ プログラムを実行する。入力データを白黒画像、種画像、出力画像の順に入力する。入力した白黒画像、種画像が存在しない場合は実行時にエラーが表示され、画像着色を行わない。

④ 着色結果(図 7:左)。納得がいかなければ種画像を編集する(図 7:右)。

図 7：実行例 2.

⑤ ③、④を繰り返して、納得できる画像を作る(図 8)。

図 8：着色結果.

4. 結果

このプログラムで着色した画像例を図 9、図 10 および図 11 に示す。それぞれ左上が白黒画像、右上が種画像、左下が着色画像である。

図 9：実行結果 1(左上:白黒画像 [10]、右上:種画像、左下:着色画像).

図 10：実行結果例 1(左上:白黒画像 [5]、右上:種画像、左下:着色画像).

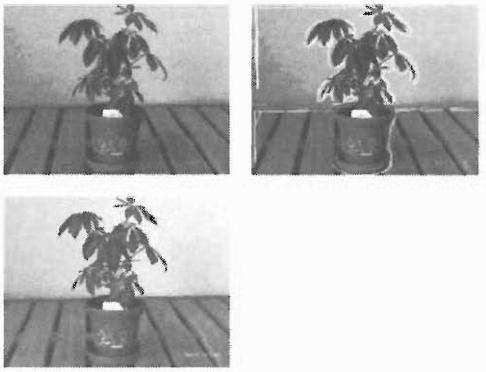

図 11：実行結果例 2 (左上:白黒画像、右上:種画像、左下:着色画像).

5. 考察

画像が大きい程、処理時間が掛かる(図 12)。また種画像に多く描画すればする程、プログラムで色を探す手間がなくなるので結果的に速く処理される(図 13)。種画像の描画において、境界線付近に描画すれば境界線を塗り分けた着色が可能である(図 14)。

図 12：画像サイズによる処理時間の比較.

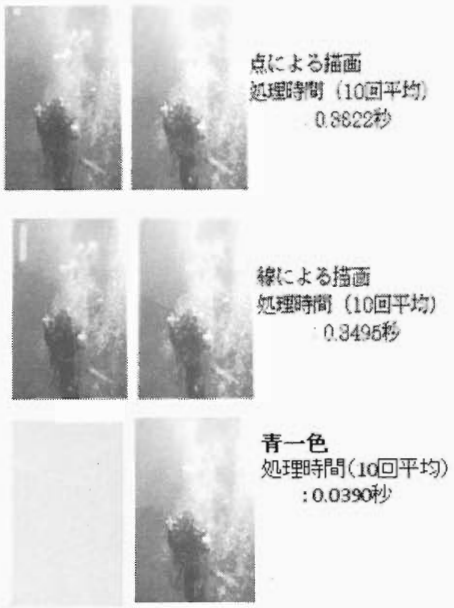

図 13：種画像の描画による処理時間.

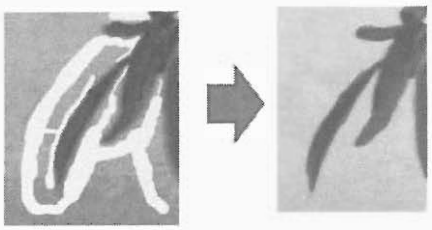

図 14：境界線付近の描画.

だが以下の問題点がある。

- 扱えるデータが BMP 形式（24bit カラー）だけである。
- 種画像を作る必要がある。本プログラムには種画像を作る機能を持たないため、別の画像処理ソフトで作成する必要がある。
- 種画像に描画する際、なるべく左上に描画しないと着色できてない箇所ができる。
- セピア等の白黒でない画像の着色が上手くできない。
- 砂利、草むら等白黒の差が大きい場所を着色するには、種画像に細かく描画する必要がある。これは閾値判定の影響によるものである。
- 大きいサイズの画像に対しての着色の処理時間が掛かりすぎる。画像サイズに応じて探索範囲を変える必要がある。

6. 終わりに

本論文では、カラリゼーションに関する基礎的なプログラミングを紹介した。「着色」だけに関しては無難に処理されている。ただし、白黒画像のカラー化による「復元」に関しては改善点が多々ある。

筆者が目指す「復元」とは、実写と着色画像の見分けがつかなくすることである。現段階の出来栄えでは、「復元」しているとはまだ言えない。これには、種画像に描画された色をもっと融通の利くように処理させる必要がある(図15)。

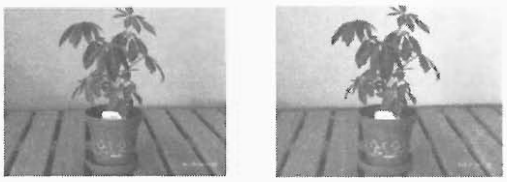

図 15：左:実写　右:着色結果.

本論文で紹介したプログラムが、今後のカラリゼーションの発展に貢献できることを期待する。

参考文献

1) 藤岡弘・中前幸治：画像処理の基礎, 昭晃堂, 2002.
2) 井上誠喜・八木伸行・林正樹・中須英輔・三谷公二・奥井誠人：C 言語で学ぶ実践画像処理, オーム社, 1999.
3) Anat Levin, Dani Lischinski, Yair Weiss "Colorization using Optimization":(http://www.cs.huji.ac.il/~yweiss/Colorization/colorization-siggraph04.pdf).
4) HSV 色空間-Wikipedia: (http://ja.wikipedia.org/wiki/HSV%E8%89%B2%E7%A9%BA%E9%96%93).
5) モノクロ画像のフリー写真: (http://tecya.net/04/monoe.html).
6) 猫時間・猫空間: (http://jumboyoshida.blog43.fc2.com/).
7) プログラミング-[物理のかぎしっぽ]: (http://hooktail.org/computer/index.php?%A5%D7%A5%ED%A5%B0%A5%E9%A5%DF%A5%F3%A5%B0).
8) RGB-Wikipedia: (http://ja.wikipedia.org/wiki/RGB).
9) 写真-Wikipedia: (http://ja.wikipedia.org/wiki/%E5%86%99%E7%9C%9F).
10) 写真共有サイト「フォト蔵」: (http://photozou.jp/photo/show/79815/37867801).