

報 文

色相分析ソフトの開発とその応用

高 橋 勝 美*¹

Development and Application of Hues Analysis Software

Katsumi TAKAHASHI*¹

Technology to extract a color of the same hue from an image on a computer as a basic technology to estimate the nutritive value of a photograph of cooking is important. However, an application to photograph cooking cannot under the present conditions, including extraction and comparison software, cannot identify a specific color. Therefore I developed software to treat a similar hue as a color of a similar shade by keeping the width to a central angle of a vector comprising the strength of three primary colors. Because it was possible to process it as a color of a similar shade from a bright part to a gloomy part with this, I was able to know what kind of ratio there was at an arbitrary point in an image and the color of a similar shade. In addition, I removed a shadow ingredient from a color and was able to emphasize the characteristic of a color in an image.

緒 言

コンピュータ上に蓄積された画像情報から特定の色を抽出することは、たとえば料理の写真からその栄養価を概算するための基礎技術として重要である。これを発展させれば、携帯電話のカメラ機能とメール機能を用いて摂食前に写真をサーバーに送信、栄養的アドバイスを受信するなどの応用も可能となり、新たなビジネスモデル構築も現実のものとなる。しかしながら現状では、特定の色の抽出や比較ソフトがゲーム開発目的に散見される程度で、料理への適用は難しい。それは、それらのソフトが純粋な1点の色のみを取り扱い、光の当たった明るい色から陰の暗い色までを同一に取り扱う同系色の概念を取り入れていないからである。

以上のことから、本研究では画像の1点からサンプリングした色情報を基にその同系色すべてを抽出するための方法を具体的なプログラムとして開発した。また、色情報の性質を利用して元の画像から陰影成分を取り去った二色差分画像を生成し、その応用について言及した。

方 法

色の表現方法¹⁾として、補色系のCMYK、純色系のRGBなどがあるが、本研究はコンピュータ上での取り扱いやすさからRGB表記(図1)を用いて遂行した。

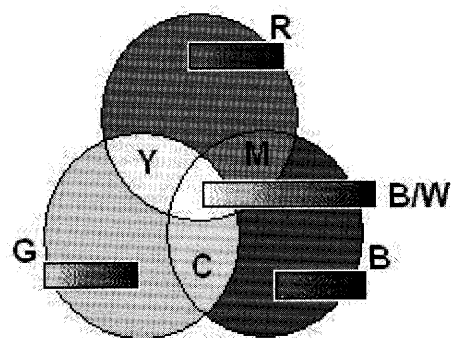


図1. RGB表記とCMYK表記

1) 同系色

RGB三原色の強度から合成されるコンピュータスクリーン上の色は図2のようにベクトルで表すことができる。RGB各原色を120°ずつ開いた軸上に配置する。図2では赤を例としたが、円周に近いほど強度は強く、中心に近いほど弱くなる。図2からわかるようにこの強度は明るさでもあり、鮮やかさでもある。単色のベクトルOR、OG、OBを合成すると混合色OSが生成される。点Sが合成色である。このとき、角 θ は色相を表し、直線OSを円周まで延長した直線上の点が表示色は同系色と言える。しかし、この直線上の点だけではなく $\theta \pm$ 数度の範囲に入る全ての直線上の点を同系色とする方が実

*¹ 情報科学研究室, 女子栄養大学: Laboratory of Information Science, Kagawa Nutrition University.

際である。本ソフトではその度数を何度にするか調整できるようにし、その角度を判定幅と定義した。

2) 彩度

軸上では円周に近づくほど鮮やかになるわけだから、直線 OS の長さ L は彩度を示すことになり、定法通り 0 ~ 255 (10進法) の数値で表現した。

3) 明度

RGB 表記では、最も明るい赤を 16 進表記で 0000FF、最も明るい緑を 00FF00、最も明るい青を FF0000 で表し、これらは図 2 の円周上に位置する。2 色混合された色であるシアン (FFFF00)、マゼンタ (FF00FF)、イエロー (00FFFF) なども円周上に位置し、その明るさは同じである。ところがこれらに第 3 色が混合されるとその分だけ

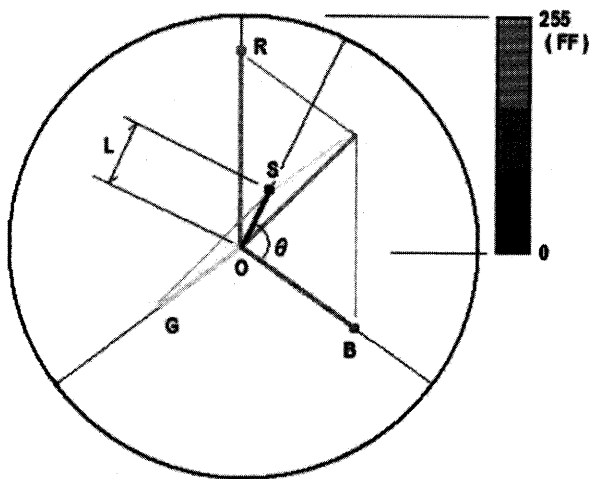


図 2. 色相環

白味が増加し、最終的に最も明るい色である白 (FFFFFF) になる。逆に、混合色の中で最も多い原色成分を第 1 色、最も少ない原色成分を第 3 色、中間の原色成分を第 2 色とすれば、第 1 色が減少すると暗くなり、最終的に黒 (000000) になっていく。このことから明るさに関係するのは第 1 色と第 3 色であることがわかる。第 1 色が 00 から FF に向かって増加するとそれとともに色は明るさを増す。第 3 色が 00 から FF (10進法で 255) に向かって増加するとその色は白さを増していき、彩度は減少するものの明度は増加する。このように、明度は最も暗い黒に始まり、彩度とともに増加する。彩度が頂点に達すると、それ以降、彩度の減少と反対に増加を続け、白となる。彩度の頂点がちょうど明度の中間値 128 (10進法) となるようにすれば最高明度、白の明度は 255 (10進法) となる。各原色は 0 ~ 255 (10進法) の値をとるから、明度は下式のように、3 原色の最高成分値と最小成分値の平均で求めた。

$$\text{明度} = (\text{MAX} (R, G, B) + \text{MIN} (R, G, B)) / 2$$

4) プログラミング

プログラミングにあたってはパーソナルコンピュータ用の市販汎用 OS である Windows 2000 Professional¹²⁾ と、その上で稼働する開発言語 Delphi ver.3³⁻⁵⁾ を用いた。

結果および考察

実際のプログラミングでは画像ファイルの検索、読み込み、表示、保存、その他、プログラムとして機能させるための付随的機能が不可欠だが、ここではそれらを省略し、同系色判定に関わる部分だけを表 1 に示した。

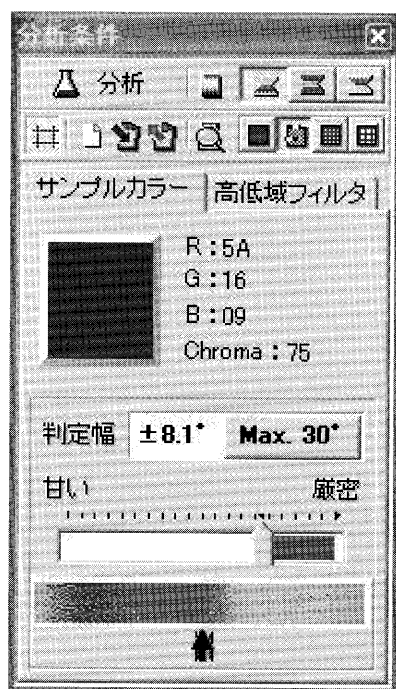


図 4. 分析条件設定

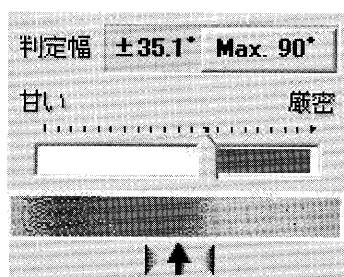


図 5. 色相バー

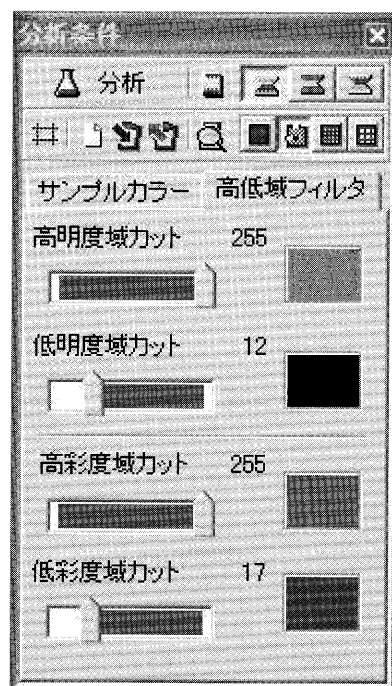


図 6. フィルタ

表 1. 同系色判定プロシージャ

```

procedure TForm1. ImgMouseDown (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  BR, BG, BB, BL, COL: LongInt;
  Cons, LX, LY: Real;
  BT, DTX: Real;
  CR: array[0..2] of LongInt;
  STX, STY: Integer;
begin
  // -----
  // 色の採取
  // -----
  if SpeedButton10. Down=False then
    begin
      Cons:=Sqrt(3)/2;
      // 色サンプル
      Panel13. Color:=Img[TabControl1. TabIndex]. Canvas.
        pixels[Round(X*OW/Img[TabControl1. TabIndex]. Width), Round(Y*OH/Img[TabControl1. TabIndex]. Height)];
      COL:=Panel13. Color;
      // 画像中のクリックした任意点の色を Panel3 に
      // 表示, その色情報を変数COLに代入
      BR:=StrToInt('$'+Copy(IntToHex(COL, 6), 5, 2)); // R
      BG:=StrToInt('$'+Copy(IntToHex(COL, 6), 3, 2)); // G
      BB:=StrToInt('$'+Copy(IntToHex(COL, 6), 1, 2)); // B
      // 変数 COL の色情報から RGB 各原色の情報を// 抽出し, 変数 BR, BG, BB に代入
      // 彩度計算
      // 色相環を棒状に表示したとき, 測定点の色相が// 端に行かないように 3 種類の開始点を設ける
      Case HUE of
        1: begin // 水色起点・赤色中心
            LX:=(BG-BB)*Cons;
            LY:=BR-(BG+BB)/2;
          end;
        2: begin // 紫色起点・緑色中心
            LX:=(BB-BR)*Cons;
            LY:=BG-(BR+BB)/2;
          end;
        3: begin // 黄色起点・青色中心
            LX:=(BR-BG)*Cons;
            LY:=BB-(BG+BR)/2;
          end;
      end;
      BL:=Round(Sqrt(Sqr(LX)+Sqr(LY)));
      // 彩度計算
      Label4. Caption:=IntToStr(BL);
      // 彩度表示
      // -----
      // チャート表示
      // -----
      Label14. visible:=False;
      DT:=Form1. TrackBar1. Position;
      DTX:=(Pi/DTM)*(100-DT)/100;
      BT:=ArcTan2(LX, LY)+Pi;
      If BT-DTX<0 then
        STX:=Round((2*Pi+(BT-DTX))*100/(628/165))
      else STX:=Round((BT-DTX)*100/(628/165));
      if BT+DTX>2*Pi then
        STY:=Round(((BT+DTX)-2*Pi)*100/(628/165))
      else STY:=Round((BT+DTX)*100/(628/165));
      // -----
      // モノクロ自動判定
      // -----
      CR[0]:=BR;
      CR[1]:=BG;
      CR[2]:=BB;
      Hantei(Sender,
        ((MaxIntValue(CR)+MinIntValue(CR)) div 2));
      if BL>TSAI then ColSet(Sender)
      else MonoSet(Sender);
      // 判定ルーチンで有彩色・無彩色を判定
      // それぞれ専用のサンプル表示を行う
      // 高彩度
      // 測定点の彩度に合わせて測定条件のフィルタ値を
      // 自動調整 (手動変更は別ルーチンで任意に可能)
      TrackBar4. Min:=BL;
      TrackBar4. Max:=255;
      TrackBar4. Frequency:=1;
      TrackBar4. LineSize:=1;
      TrackBar4. PageSize:=10;
      TrackBar4. SelStart:=TrackBar4. Min;
      TrackBar4. SelEnd:=TrackBar4. Max;
      TrackBar4. Position:=TrackBar4. Max;
      TrackBar4Change(Sender); // フィルタ色指定
      if BL=255 then

```

```

begin
    TrackBar4. Enabled:=False;
    Label9. Enabled:=False;
end
else
begin
    TrackBar4. Enabled:=True;
    Label9. Enabled:=True;
end;
// 低彩度
// 測定点の彩度に合わせて測定条件のフィルタ値を
// 自動調整（手動変更は別ルーチンで任意に可能）
TrackBar5. Max:=BL;
TrackBar5. Min:=0;
TrackBar5. Frequency:=1;
TrackBar5. LineSize:=1;
TrackBar5. PageSize:=10;
TrackBar5. SelStart:=TrackBar5. Min;
TrackBar5. SelEnd:=TrackBar5. Max;
TrackBar5. Position:=TrackBar5. Min;
TrackBar5Change(Sender); // フィルタ色指定
if BL=0 then
begin
    TrackBar5. Enabled:=False;
    Label10. Enabled:=False;
end
else
begin
    TrackBar5. Enabled:=True;
    Label10. Enabled:=True;
end;
// 高明度
// 測定点の明度に合わせて測定条件のフィルタ値を
// 自動調整（手動変更は別ルーチンで任意に可能）
CR[0]:=BR;
CR[1]:=BG;
CR[2]:=BB;
TrackBar2. Min:=(MaxIntValue(CR) + MinIntValue(CR)) div 2;
TrackBar2. Max:=255;
TrackBar2. Frequency:=1;
TrackBar2. LineSize:=1;
TrackBar2. PageSize:=10;
TrackBar2. SelStart:=TrackBar2. Min;
TrackBar2. SelEnd:=TrackBar2. Max;
TrackBar2. Position:=TrackBar2. Max;
TrackBar2Change(Sender); // フィルタ色指定
if TrackBar2. Min=TrackBar2. Max then
begin
    TrackBar2. Enabled:=False;
    Label1. Enabled:=False;
end
else
begin
    TrackBar2. Enabled:=True;
    Label1. Enabled:=True;
end;
// 低明度
// 測定点の明度に合わせて測定条件のフィルタ値を
// 自動調整（手動変更は別ルーチンで任意に可能）
CR[0]:=BR;
CR[1]:=BG;
CR[2]:=BB;
TrackBar3. Max:=(MaxIntValue(CR) + MinIntValue(CR)) div 2;
TrackBar3. Min:=0;
TrackBar3. Frequency:=1;
TrackBar3. LineSize:=1;
TrackBar3. PageSize:=10;
TrackBar3. SelStart:=TrackBar3. Min;
TrackBar3. SelEnd:=TrackBar3. Max;
TrackBar3. Position:=TrackBar3. Min;
TrackBar3Change(Sender); // フィルタ色指定
if TrackBar3. Min=TrackBar3. Max then
begin
    TrackBar3. Enabled:=False;
    Label2. Enabled:=False;
end
else
begin
    TrackBar3. Enabled:=True;
    Label2. Enabled:=True;
end;
end;
end;
end;

```

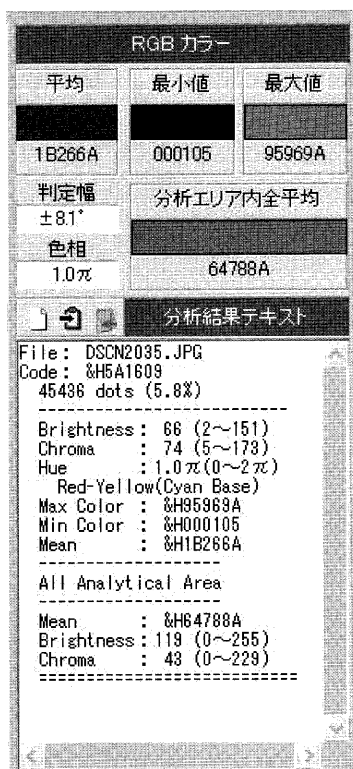


図 7. 分析結果

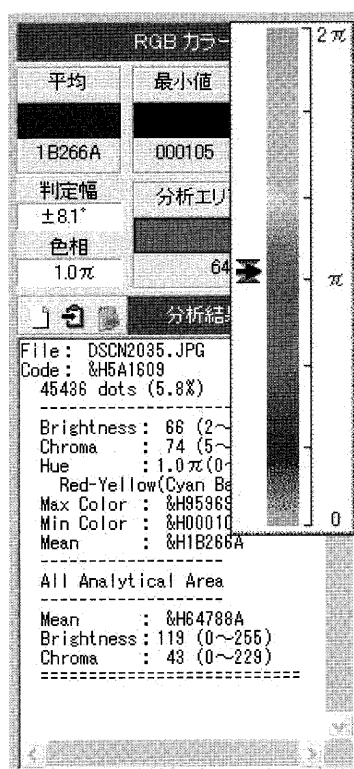


図 8. 分析結果 (色相)

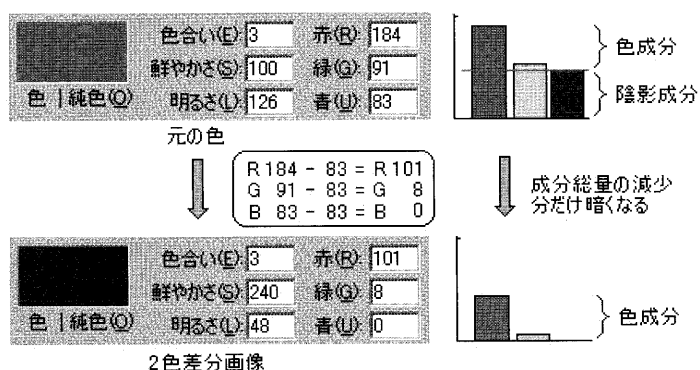


図10. 二色差分画像の原理

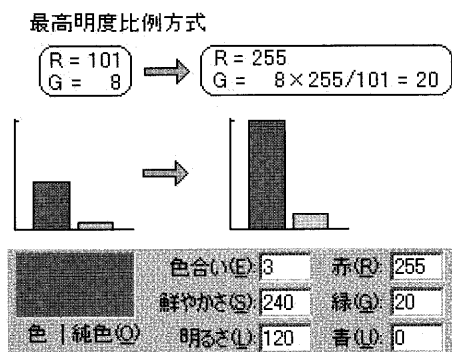
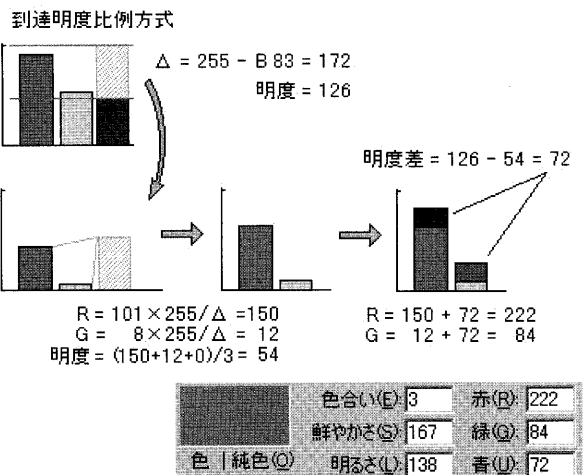
図11. 二色差分画像における明度補正
(最高明度比例方式)

図12. 二色差分画像における明度補正 (到達明度比例方式)

1) サンプリング

画面上に表示させた画像の1点を指定することにより、その点の色と同系色をすべて抽出させる。点の指定は画像上の目的位置でマウスをクリックすることで行い、同時にその点の色情報を取得、無彩色であれば判定対象外、有彩色であれば同系色判定する際の明度、彩度範囲外をカットするフィルターの調整などの計算を実行させた。画像上でマウスのボタンが押された際のプログラムからサンプリングに関わる部分を表1に記し、このプロセスと付随プロセスによって実現する画面を図3に示した。画面右下の分析条件欄には指定した点のRGB値、彩度(chroma)、実際の色、などの情報が示されている。

2) 分析条件の設定

そのままサンプリングした色の同系色を抽出することが可能だが、分析精度、判定幅、色相バリエーション、カットフィルターなどの設定を事前に行うことでより目的に合致する抽出がおこなえるようにした。(図4)。

2-1) 分析精度

画像の左上を原点として、1点ごとにサンプリングした色の同系色範囲に入るかどうか走査することは精密な結果が得られる反面、画像の大きさにともなって時間を要し、それほどの精度を求めない場合は非効率である。また、自然な画像では輪郭部分以外は隣り合う点の色情報が極端に変化することは少ない。したがって高精細～超高速まで、走査間隔を指定できるようにした。

2-2) 判定幅

既述のように、図2において点Sをサンプル色とすれば、直線OSとその延長、および両脇の領域が同系色の範囲となる。この両脇の範囲を中心

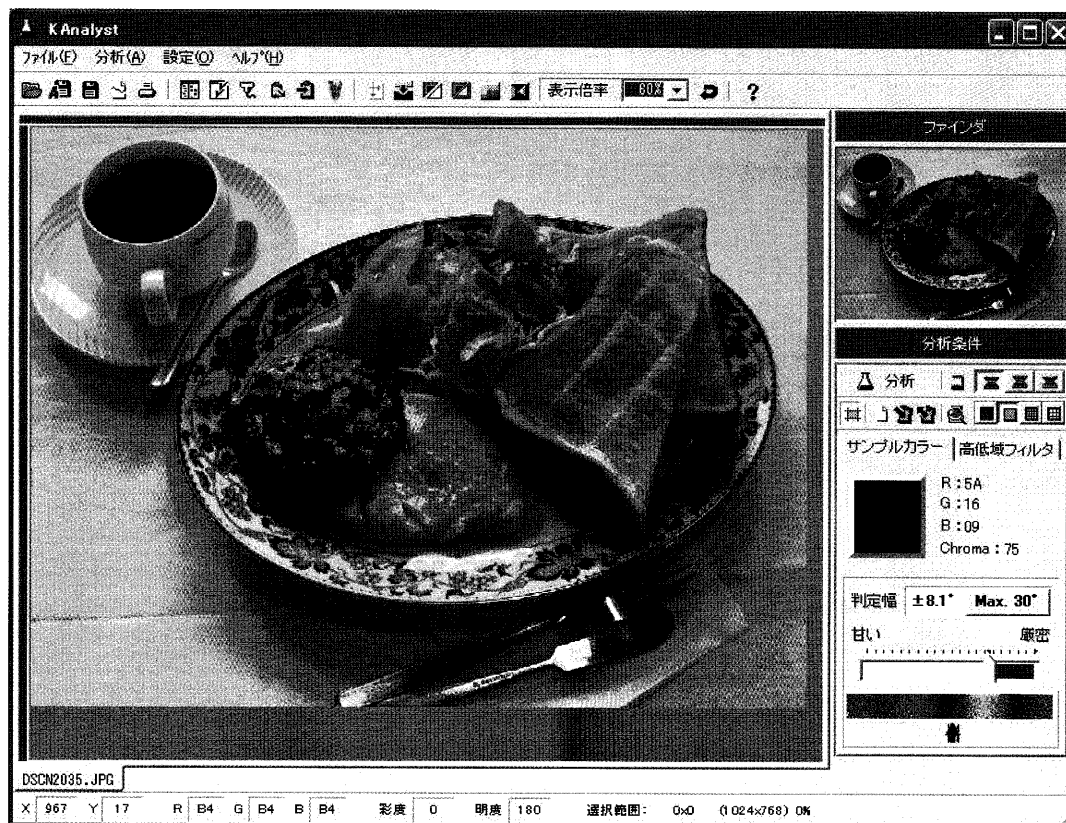


図 3. サンプルング画面

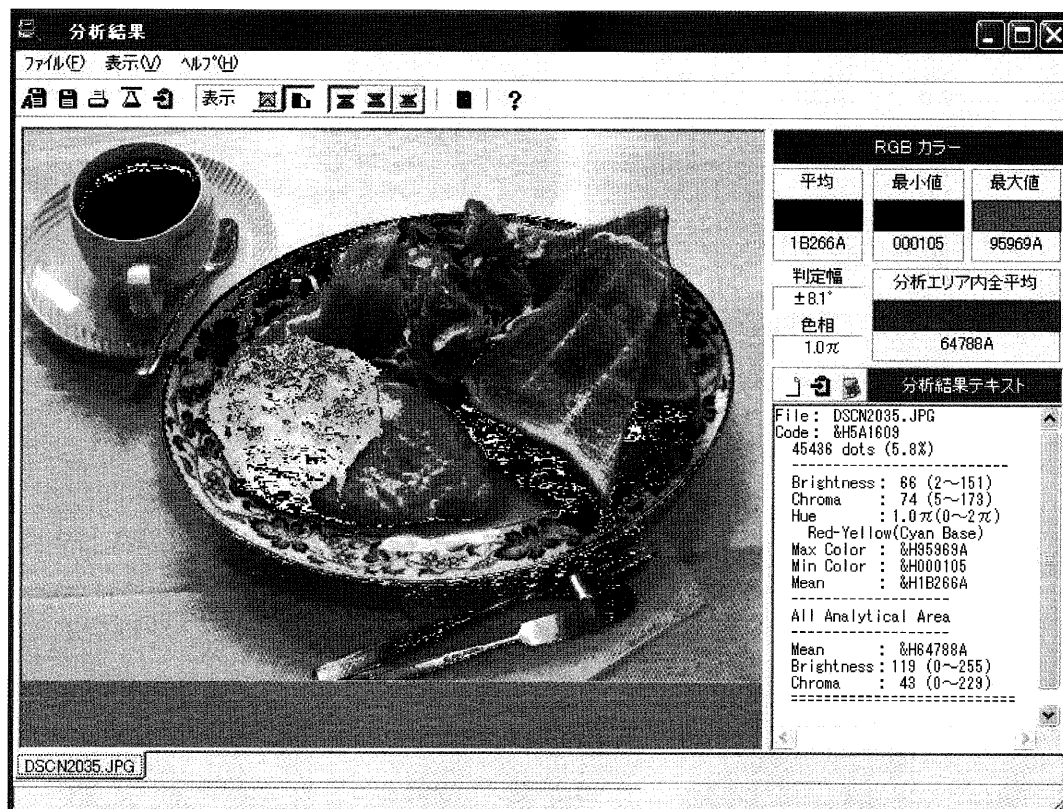


図 9. 分析範囲確認 (色反転部分)

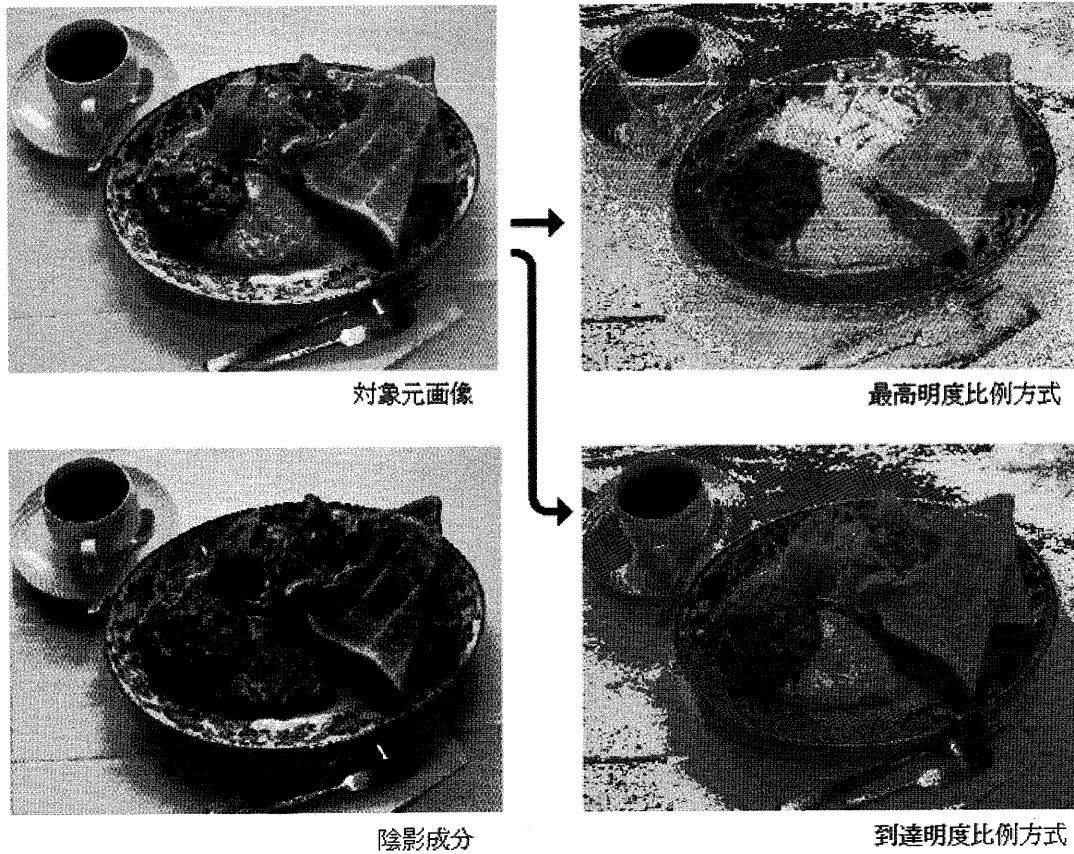


図13. 明度補正方式の違いによるふたつの二色差分画像

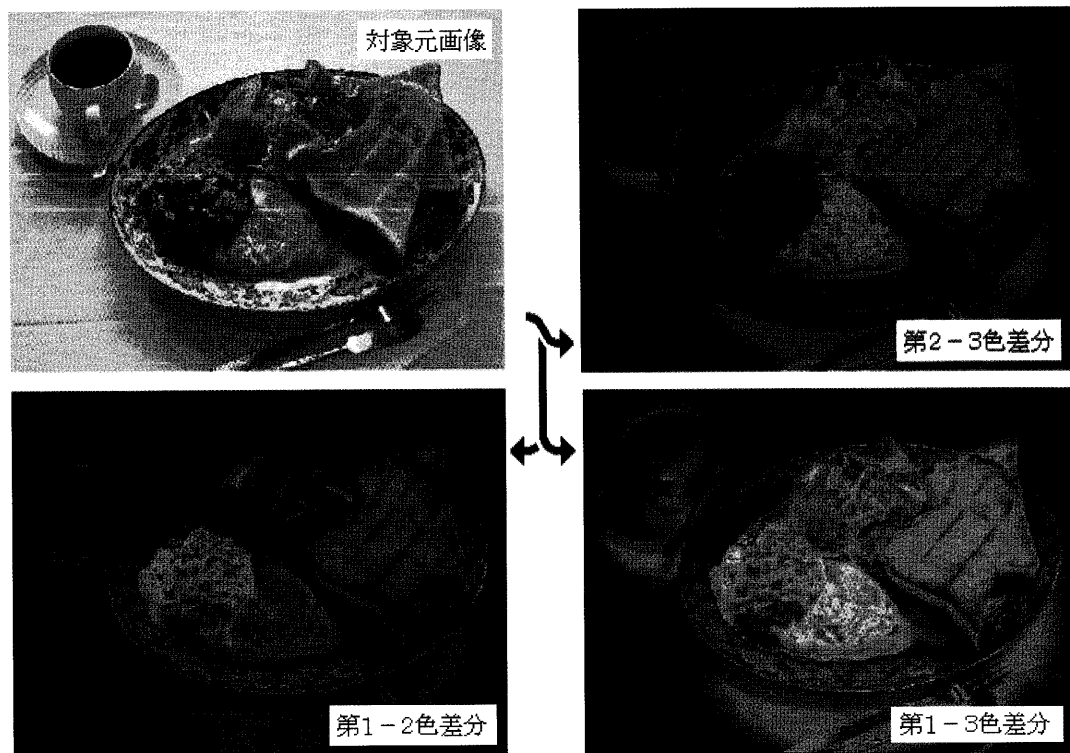


図14. 二色間の差分による画像

角で左右何度まで認めるかが判定幅であり、当初、最大 $\pm 30^\circ$ としたが、運用の結果最大 $\pm 90^\circ$ とするのが妥当であった。

2-3) 色相バー起点

図2の色相環を棒状に開いたバー(図5)を考えるとプログラム上の取り扱いが簡便である。しかし、サンプル色によってはバーの両端に2分され、不都合が生じる。そこでサンプル色に応じてバーの起点色を適宜変更し、サンプル色がなるべくバーの中心に位置するようにする必要があった。この変更をボタンで容易に実現できるようにした。

2-4) フィルタ

どの色相であれ明度が増加していけば白になり、明度を減じていけば黒になる。写真ではこの現象はしばしばあり、特にデジタルカメラの撮像素子は明るさに弱く、白飛びと呼ばれる色データ飽和現象を起こす。こうした真っ白部分、あるいは真っ黒部分を測定範囲に含めるか否かは目的によっては重大事である。そこで本ソフトでは任意の低明度域～高明度域、高彩度域～低彩度域の範囲内だけを測定範囲とし、それ以外をカットするためのフィルタを具備した(図6)。サンプル色は必ずこの範囲内に存在しなければならないため、既述のサンプリングプロシージャにはサンプリング時にフィルタとの整合性を判断し、サンプル色が範囲外にあればフィルタ範囲を補正する仕様とした。

3) 条件合致部分の抽出

「分析」ボタン(図4)をクリックにより、対象の点が前項で設定された色相、判定幅、フィルタ範囲内にあるかどうか画像全体を走査し、該当する点ごとの色情報と数を蓄積、画像全体に占める割合や平均の明度、彩度などの分析結果をテキストとして表示させた。また、それらを実際の色としても表示させた(図7, 8)。

4) 条件合致部分の確認

実際にどの点が条件範囲内に該当しているのかが視認できないと、たとえばパンの焼けこげとソースが同じ色範囲として判定され、思わぬ誤りが起こり得る。本ソフトは単に色情報だけを扱っているので現状ではこうしたことは不可避である。将来的には面情報を考慮してテクスチャの違いから物質を切り分ける方法も考えなければならない。現状で正確な分析を遂行する便法として、条件に該当する部分を反転表示させて分析者に確認させ、結果が満足できない場合は条件を変えて再分析できるようにして誤りを回避した。

以上述べたサンプリング、分析条件設定、結果の確認を中心機能とし、明度、彩度、色相それぞれの走査グラフ表示、等階調表示、輪郭抽出などの付随機能によって

本ソフトは構成されている。これを用いた料理写真の色分析に向けて現在実用化テスト中である。

なお、一般に、明度はR, G, Bの最高成分値^{6,7)}とされており、本研究においても当初はこれに準じた処理を行っていた。しかし、写真で運用してみると得られる数値と写真を見た感覚とにずれがあり、実際的ではなかった。そこで感覚と一致し、理論上も問題の無い計算式として、方法で述べたように最高成分値と最小成分値の平均を用いた。この方法は本研究独自のものであり、さらに多くの事例を集めて妥当性を検証する必要がある。しかし、本ソフトで複数の写真について得られた結果を比較する限り、計算方法の違いは問題とならない。

5) 二色差分画像への応用

コンピュータ上ではすべての色は赤(R), 緑(G), 青(B)の3原色から構成されているのは前述の通りであり、その値は0～255(10進表記)の数値をとる。したがって表示できる限界は $256 \times 256 \times 256 = 16777216$ 色となる。ある1点のRGB値に注目してみる。ある写真からサンプリングした色、 $R=184, G=91, B=83$ を考えると、RGBに共通に含まれる色、 $(R, G, B) = (83, 83, 83)$ は3原色が同じ値だからモノクローム(白～灰～黒色:無彩色), すなわち、陰影成分である。色情報からこの陰影成分を除く、すなわち、

$$(R, G, B) = (184-83, 91-83, 83-83)$$

を求めると、再構成される画像は陰影を除いた純粋な色の成分だけで構成されているはずである。この画像を二色差分画像と命名した(図10)。本ソフトのプロシージャを呼び出すことにより、容易に二色差分画像を生成することが出来る。ところが、1色減じた分だけ明度が低下するため、これを補正し、元の画像の明度に近い二色差分画像を得ることが必要である。明度補正の方法として、ふたつの方法を考案した。

5-1) 最高明度比例方式

二色差分,

$$(R, G, B) = (184-83, 91-83, 83-83) = (101, 8, 0)$$

の最大成分である赤を、101から、とりうる最大値である255にし、残る緑を比例配分すれば明度は元の明度に近いものとなる。この方法では各点の持つ最大成分(この例では赤)が特に強調され、立体感はまったく失われる(図11)。

5-2) 到達明度比例方式

はじめに元の色,

$$(R, G, B) = (184, 91, 83)$$

と二色差分,

$$(R, G, B) = (101, 8, 0)$$

との明度差を計算しておく。次いで元の色(の最小成分 $B=83$)と、とりうる最大値255との差、172を求める。この172を基準にすると

$$(R, G, B) = (150, 12, 0)$$

となる。これに明度差 72 を加えると

$$(R, G, B) = (150 + 72, 12 + 72, 0 + 72) = (222, 84, 72)$$

となる (図 12)。この方式だと二色差分と称しながら、明度差の加算により 3 色とも 0 を越える値を持つから、より自然な画像が得られる。それでもなお元の画像よりは色の特徴をつかみやすい。

これらの計算を行うにあたって、対象元画像で無彩色の点は計算から除外、そのまま画像を構成している。したがって真っ黒部分などはそのまま残っている (図 13)。

5-3) 二色差分画像応用の可能性

元の分析対象画像を前処理として二色差分画像に再構成し、その後同系色判定を行えば、各部の色の特徴が強調されているためにより明快な分析結果を得られるものと期待できる。また、この計算の応用として、RGB の最大成分を第 1 色、最小成分を第 3 色、中間成分を第 2 色とし、1~2, 1~3, 2~3 色それぞれの差分を求めることもできる。画像全体では RGB のどれが第 1 色、第 2 色、第 3 色になるかが点ごとに異なるため、カラー表示しても意味のないことが多い。そこでこれらの差分から求めた明度に基づいてモノクローム画像を再構成する (図 14) ことにより、人物の肌色で目立たない静脈の青色を浮き立たせることや、コンピュータ上で合成されたアイドルコラージュなどの頭部と体部の接合境界線を見つけることなどが可能であった。現在、金属材料やメタル製品検査機器と本ソフトの連動を検討中である。これらの応用例から、食品、調理分野でも写真から食品材料などの物質的切り分けが容易におこなえるものと期待される。

以上のことから、料理の写真をコンピュータに取り込み、二色差分画像化などの前処理を経て、同系色ごとの専有面積比を求め、栄養価と対にしたデータを蓄積することが可能になる。将来的には、同系色範囲内の明度と彩度の振れを計測すれば表面のテクスチャをおおまかに

把握できるから、食材ごとに切り分けてのデータ蓄積が期待できる。本ソフト開発により、栄養価未知のサンプル画像をこれらの蓄積データと比較することで栄養価を類推すること、すなわち、写真による栄養価自動計算の可能性が示唆された。

要 約

料理の写真からその栄養価を概算するための基礎技術として、画像から近似する色相を、明度、彩度の幅を持たせて抽出、面積割合を算出する応用ソフトを開発した。開発にあたり、実際の写真がもつ感覚的な明るさと計算値の整合性を考慮し、三原色の最高成分値と最小成分値の平均を明度とした。これにより、画像中の任意の点とその同系色がどのような割合で存在するかを知ることができた。また、画像を構成する各点の色情報から陰影成分を除去、明るさを補正したうえで再構成することにより、色彩の特徴を強調した二色差分画像を定義した。二色差分画像を用いると、色彩の微妙な変化に影響を受けないために、食品材料などの物質的な切り分けが容易になるものと期待された。

文 献

- 1) 応用物理学会 光学懇話会 編: 色の性質と技術. 朝倉書店, 56-71 (1990)
- 2) Microsoft 社: Microsoft Windows 2000 Professional ファーストステップガイド
- 3) Borland International: Delphi コンポーネント開発者ガイド. ボーランド㈱ (1995)
- 4) Inprise Corporation: Borland Delphi 5 開発者ガイド. インプライズ㈱ (1999)
- 5) Borland International: Turbo Pascal 6.0 プログラマーズガイド. ㈱ボーランドジャパン (1990)
- 6) 高木幹雄, 下田陽久: 新編 画像解析ブック. 東京大学出版会 (2004)
- 7) 石立 喬: VisualC++6.0 を用いた易しい画像処理 (3). http://homepage3.nifty.com/ishidate/vcpp6_g3/vcpp6_g3.htm (2002)