2016年度 卒業論文

パーソナルカラー診断アプリケーションの開発

大阪産業大学 デザイン工学部 情報システム学科
情報教育システム研究室

13H027 亀尾菜月

目次

1 **はじめに** 1

2 **目的** 2

3 **色について** 3
- 3.1 似合うとは . . . 3
- 3.2 似合う色と好きな色 . . . 3

4 **パーソナルカラー** 4
- 4.1 フォーシーズンタイプ . . . 4

5 **従来の診断方法** 7

6 **パーソナルカラー診断の Web アプリケーション** 8
- 6.1 動作 . . . 8
- 6.2 写真の撮影条件 . . . 9
- 6.3 OpenCV . . . 10
- 6.4 Flask . . . 11
- 6.5 顔の検出及び目・鼻・口の各パーツ検出 . . . 11
- 6.6 フォーシーズンの判定方法 . . . 11

7 **アンケート調査** 13

8 **結果と考察** 14
- 8.1 アンケート結果 . . . 14
- 8.2 考察 . . . 15
- 8.3 今後の課題 . . . 16

9 **結論** 17

付録 A ソースコード 19
- A.1 パーツ検出 . . . 19
- A.2 パーソナルカラー判別 . . . 20
- A.3 Web アプリケーションとしての動作部分 . . . 23

1　はじめに

私たちが新しい服を買う、髪を染めるなど、いままでの自分とは異なるスタイルになろうとしたときに、流行色やファッション雑誌を参考にすることがあるだろう。ところがいざ身にまとうと、自分に似合う色かという疑問がある。もし自分に似合わない色を選択すると、垢抜けないあるいは顔がくすんで見えてしまう。そこで、自分を一番きれいに見せるために、自分に似合う色とされるパーソナルカラーを把握する必要がある。

第 2 章では本研究の目的について述べる。第 3 章では色について述べる。第 4 章ではパーソナルカラーについて述べる。第 5 章では従来の診断方法について述べる。第 6 章では本研究で開発したシステムの概要について述べる。第 7 章では Web アプリケーションのアンケート調査について述べる。第 8 章ではアンケートの結果をその考察とともに述べる。第 9 章には研究の成果とともに今後の課題についてまとめる。

2　目的

パーソナルカラーの診断方法として、資格を持つ人に診断してもらう方法と Web サイトで自分で診断する方法がある。ところが前者の方法では、診断者の経験や感性によって結果が変わるという問題があり、後者の方法では、自己診断が難しい設問があり、客観的な意見が必要という問題があった。これを解決するためには、統一した基準でパーソナルカラー診断をできるシステムが望ましいと考えた。そこで本研究では、客観的な意見を必要とせずより手軽な診断をすることを目的とし、Web アプリケーションの開発を行った。

3　色について

色というのは、赤色でも青みがかった赤であったり黄みがかった赤があるように、ほとんどの色が青色寄りか黄色寄りかのどちらかにわけられる [1]。そして、生まれ持った髪や肌などの色が青寄りか黄色寄りか調べ、より多い方でブルーベースかイエローベースかが決まる。イエローベースは色相が黄みに寄っている色達の総称で、全体的に暖かくカラフルな色味を持つ。ブルーベースは色相が青みに寄っている色達の総称で、全体的に冷たくて澄んでいるイメージの色味を持つ。

3.1　似合うとは

似合うとは自分が持つ色や雰囲気につりあっている状態、調和している状態のことをいう。回りの色によって見え方が異なる様子を図 1 に示す。

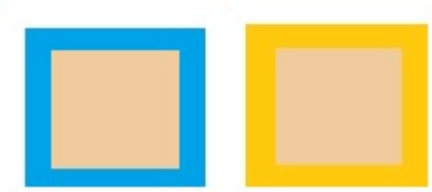

図 1　肌の見え方が左右で違う。左は肌と回りのメリハリがある。右は色がぼやけて見える。

図 1 の左側は色の境目がはっきりして肌色がきれいにみえる。図 1 の右側は色がぼやけていて周りの色のほうが際立つ。この左の図の状態が調和している、つまり似合っているという状態である。

似合う色を身につけることで以下のメリットが得られる。

- 肌がきれいに見える
- 若々しく、顔色がよく見える
- 安いものを身に着けても上質な雰囲気になる

逆に似合っていない色を身につけると以下のデメリットが生まれる。

- 老けて見える
- 疲れて見える
- センスが悪く見える
- 欠点が目立つ
- 不自然に見える

3.2　似合う色と好きな色

よく、似合っていると誤解しがちなのが自分の「好きな色」である。好きな色は主観的な要素を強く持ち、似合っていると思い込みがちになる。だからといって好きな色を身につけることは悪いことではない。好きな色には、自分の心を癒やしてくれたり、気分をよくしてくれるというメリットがあるからである。つまり、第三者の目を気にしなくてはいけない場面とそうでない場面を使い分けることが大切である。好きな色が自分に似合う色であるのが 1 番いいのだが、もし似合わない色であってもインテリアなどに取り入れることができる [2]。

4　パーソナルカラー

パーソナルカラーとは、人が生まれ持った色の魅力を最大限に引き出す色のことである。つまり、自分に似合う色のことである。

パーソナルカラーにはいくつかグループが存在する。一番最小のグループがブルーベースとイエローベースの 2 つに分けたものだ。ブルーベースとイエローベースを明度や彩度でそれぞれ 2 つに分けると、フォーシーズンタイプという 4 つのタイプに分かれる。各グループの色調が四季を連想させ、春・夏・秋・冬の各タイプに特徴がある。

4.1　フォーシーズンタイプ

- スプリングタイプ

イエローベースで温かみがあってカラフルで明るい色が似合う。
若々しい色や多色使いをしてもかわいく着こなすことができる。
キーワード：キュート、ポップ、若々しい
明度：高い色が多い
彩度：高い色が多い
似合うファッションを図 2、パーソナルカラー例を図 3 に示す。

図 2　スプリングタイプのファッション例。左はシンプルにまとめて上品なイメージ。真ん中は華やかな柄を使い、大人っぽいイメージ。右はがーリーの中にカジュアルなイメージ [3]。

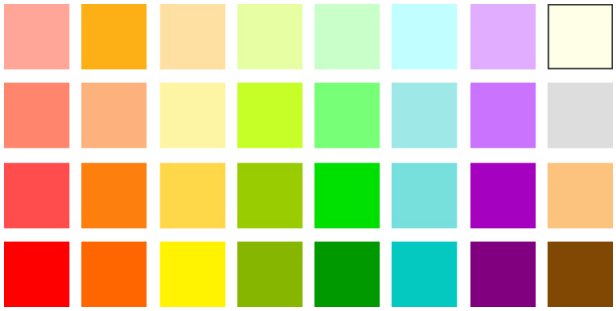

図 3　スプリングタイプのパーソナルカラー例

- サマータイプ

ブルーベースでソフトで優しく、涼しさがあるくすんだ色が似合うタイプ。
パステルカラーや柔らかい生地を身にまとうとよりおしゃれな雰囲気になる。
キーワード：優しい、エレガント、さわやか
明度：高い色が多い
彩度：低い色が多い
似合うファッションを図 4、パーソナルカラー例を図 5 に示す。

図 4　サマータイプのファッション例。左は王道のドレススタイル。真ん中は服を淡い色にして鞄を目立たせている。右はやわらかなタッチの柄を使い、ナチュラルなイメージ [3]。

図 5　サマータイプのパーソナルカラー例

- オータムタイプ

イエローベースでおちつきがある少しくすんだ色が似合うタイプ。
金色のアクセサリーを身につけるとゴージャス感が増す。
キーワード：シック、落ち着き、ナチュラル
明度：低い色が多い
彩度：低い色が多い
似合うファッションを図 6、パーソナルカラー例を図 7 に示す。

図 6　オータムタイプのファッション例。左は柄シャツを着て、高級なイメージ。真ん中は甘めの服を着つつ、色で大人っぽさを出している。右はスカートをメインにして、ほかを淡い色に抑えている [3]。

図 7　オータムタイプのパーソナルカラー例

- ウインタータイプ

ブルーベースで涼しげがあって鮮やかな色が似合うタイプ。
鮮やかな色でも着こなすことができる。
キーワード：シャープ、クール、個性的
明度：低い色が多い
彩度：高い色が多い
似合うファッションを図 8、パーソナルカラー例を図 9 に示す。

図 8　ウインタータイプのファッション例。左はクールなイメージ。真ん中はお嬢様をイメージ。甘くなりすぎないようにビビッドカラーをいれている。右はメンズアイテムを取り入れている [3]。

図 9　ウインタータイプのパーソナルカラー例

5　従来の診断方法

パーソナルカラー診断をする方法は主に 2 つある。1 つは専門の資格を持ったカラーリストが行う診断である。この診断にはドレープという布を使う。ドレープは大きく 4 つのグループに別れた色が使用され、例えば 4 種類の赤色の布でどの赤が一番似合うか、顔色の変化を観ていく。客観的な判断を得られるため、パーソナルカラーの中でもその人の雰囲気に合わせた色やどう見せたいかという相談をすることができる。その代わり、診断料金がかかること、診断場所に行かなければいけないこと、カラーリストとの相性があるという問題がある。

もう 1 つは Web サイトでいくつかの質問に答える診断である。髪、目、頬、肌、唇の色を答えるだけのものから数十問の質問が設置されているサイトがある。実際に Web サイトで診断をしようとすると、質問数にばらつきがあり、どの Web サイトで診断をすればいいのか迷う。質問は、肌は青みがかっているか黄色がかっているかを自分で判断しなくてはならず、初心者には答えることが難しいものや、各パーツに近い色を選ぶ形式だと、近しい色が無い、わからないという問題がある。

6　パーソナルカラー診断の Web アプリケーション

従来の Web サイトでは、複数の質問があるため回答に時間がかかったり、自分の持つ色がはっきりとわからず曖昧さが残る回答をしなくてはいけなかった。そこで、OpenCV の物体検出を利用して自分の持つ色を自動で判断した。それによって質問そのものを無くして、診断にかかる時間と手間を減らすことを試みた。また、主観的なアプローチをなくしたことで、カラーリストに診断してもらっているような客観的な意見を聞くことができる診断にした。

6.1　動作

TOP ページには図 10 に示す画像送信フォームがある。

図 10　トップページ。この画面で顔写真を送信する。

顔が写った写真を送ると、目、鼻、唇と思われる画像が表示される。画像を図 11 に示す。各画像に番号が振ってあるので、それぞれが中心に写っているものを選ぶ。該当する画像が無ければ空欄のままでも診断は続行できるが、信頼性が低くなるため違う顔写真を送り直すことが望ましい。

図 11　目、鼻、唇の選択画面

目、鼻、唇が写っている写真の番号を入力フォームに入れて送信すると結果ページが表示される。結果はベースカラーとフォーシーズンタイプの各説明とパーソナルカラーが表示される。診断はこれで終了である。

6.2　写真の撮影条件

判定には顔写真を 1 枚使用する。パーソナルカラーとは自分が生まれ持った色に似合う色を診断するものなので、できるだけ手を加えていない状態で撮影することが重要である。以下に撮影をする際の望ましい条件を示す。

- 化粧は落とした状態で撮影する
- 髪の毛は染めていない状態がよい
- 前髪が目にかからないようにする

- 顔は正面から撮影する
- 口は閉じる
- 加工写真は使用しない
- 順光で明るいところで撮影する

証明写真を取るときと同じようにポーズや表情をすると、パーツ検出がしやすくなる。ホワイトバランスの調整ができるのであれば太陽光下で撮影したように調整するとよい。そうすることで、室内で撮影した場合でも蛍光灯などの影響が減り、自然に近い色にすることができる。検証時には iPhone を使用した。縦向きで撮影すると送信時に横向きで送信され顔の検出ができなかったため、横向きで撮影しなければならない。

6.3 OpenCV

OpenCV(正式名称：OpenSourceComputerVisionLibrary) は Intel 社が開発した画像処理・画像解析、機械学習などの機能を持つライブラリである [4]。使用できる言語は C・C++、Python、Java などである。今回は OpenCV2.7.14 とそれに合わせて python2.7 を使用した。

6.3.1 **カスケード分類器**

物体の検出をするためには、検出したい物体の特徴を抽出した「特徴量」を学習データとしてまとめる必要がある。この特徴量がまとまったデータをカスケード分類器という。カスケード分類器は自分で作ることができるが、今回は OpenCV に標準搭載されている顔・目・鼻・口の検出器を利用した。

特徴量の抽出方法は haartraining である。これは、明暗で構成される短形でパターンを見ていき、物体にマッチングするかどうかを判定する Haar–Like 特徴によって機械学習行っている。

6.3.2 Haar–Like

顔画像を 2 値化してぼかすと、明暗差で目や口の位置がそれとなくわかる。同じように、機械で明暗差を検知して目や口の位置を割り出すのが Haar–Like である。明暗差で特徴量を取っているため、逆光や薄暗いところでは使えないという弱点がある。

図 12 のように黒と白の部分はいくつかパターンがあり、このパターンを組み合わせマッチングを行う。

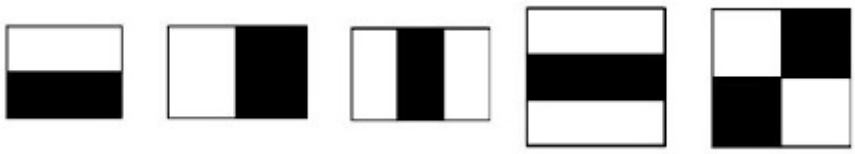

図 12　Haar–Like のマッチングパターン

パターンで明暗差を見ていき、最終的に目や唇の明暗パターンを学習していく。目であれば瞳の部分は暗い部分、まぶたや頬は白い部分になる。そのため、下図のパターンが 2 つマッチングすると目の候補になる。さらに目の回りの明暗差も取り、パターンの配置によって目を学習することができる。実際に検出するときも同様にマッチングを行い、パターンの配置があらかじめ学習していた目と同じかどうかで判断されている。パターンはサイズを変えることができ、検出時は縮小率を指定することができる。マッチングの様子を図 13 に示す。

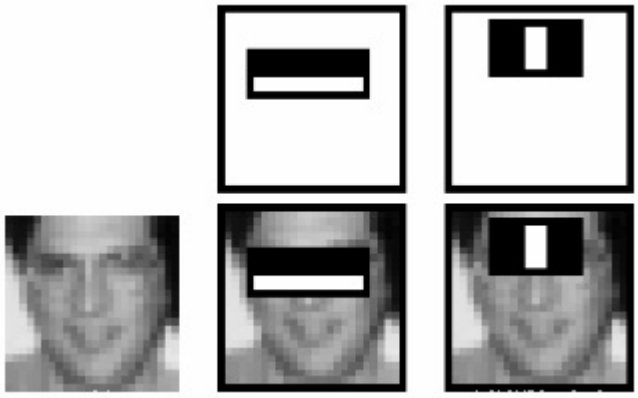

図 13　Haar–Like のパターンを使ってマッチングを行っている

6.4　Flask

Flask とは Python 用の軽量 Web 開発フレームワークである [5]。HTML テンプレートエンジンの JInja2 と Web アプリケーションと Web サーバを接続する標準仕様である WSGI が標準搭載されている Werkzeug に依存している。これを用いて、Web アプリケーションにした。

6.5　顔の検出及び目・鼻・口の各パーツ検出

ユーザが送信した画像は haar–like で学習したカスケード分類機によって、顔を含む短形が切り取られる。切り取った範囲は縦を y 軸、横を x 軸とし、それぞれ y:y+h,x:x+w とする。この範囲で各パーツの探索を行う。各パーツは顔の中に存在するため、探索箇所を減らすためである。画像は BGR で読み込むが顔検出前に 2 値化しておく。これは Haar–like が明暗差をもとに検出をしているため、2 値化することで高速に探索をするからである。

6.6　フォーシーズンの判定方法

フォーシーズンタイプの明度と彩度の特徴を利用して、診断者が生まれ持つ色をフォーシーズンタイプのどれに属するかを振り分けることができる。

Opencv の明度と彩度はそれぞれ 0∼100 の数値で表す。フォーシーズンの振り分け方を表 1 に示す。

表 1　フォーシーズンタイプの明度と彩度の数値

タイプ	明度	彩度
スプリング	50∼100	50∼100
サマー	50∼100	0∼49
オータム	0∼49	0∼49
ウインター	0∼49	50∼100

本来であれば髪、目、頬、肌、唇の色で診断することがこのましいのだが、OpenCV では髪を抽出することが難しく、日本人は頬にあまり色がでないため、今回の判定では目、肌、唇の色で行う。パーツの形状を切り出すことが困難だったため、パーツが中心に写っている画像から中心点とその回り 4 点で計 5 点の彩度と明度の平均値を算出した。上の表をもとに、彩度と明度の平均値で 4 つのタイプにわけた。

人によっては全てが同じタイプではなく、目と鼻がスプリングタイプで口がオータムタイプということもあり

うる。そのような場合は一番多く当てはまったタイプがその人のタイプとした。もし目がスプリングタイプ、鼻がオータムタイプ、唇がウインタータイプというような全てが違うタイプになったときは診断不可能とする。

6.6.1 BGR

BGR とは赤色・緑色・青色で色を表現する方法で、Blue・Green・Red の頭文字を取っている。この 3 色を混ぜ合わせるほど明るい色に変化していき、幅広い色へと再現することができる。一般的には RGB というのだが、Opencv では BGR の順で出力されるため頭文字の順番が変わっている。

6.6.2 HSV

HSV とは色相・彩度・輝度で色を表現する方法で、Hue・Saturation・Lightness の頭文字を取っている。BGR の色をそれぞれ反転させたものが HSV のである。彩度を上げると原色に近づき、下げると黒に近づく。輝度を上げると白に近づき、下げると黒に近づく。BGR を HSV に変換するとネガポジ反転している状態になる。

7　アンケート調査

開発した Web アプリケーションを実際に試してもらい、従来のシステムを使ったときと比較したアンケートを実施した。従来のシステムは比較的出題数が少ない IROUSE [6] を例に出した。被験者は女性 5 名で実施した。

アンケート内容は以下のとおりである。

- 性別
- フォーシーズンタイプの診断結果
- 診断結果は自分に当てはまると思うか
- 従来の Web 診断サイトと比べて楽か
- 従来の Web 診断サイトのフォーシーズンタイプの診断結果

8　結果と考察

8.1　アンケート結果

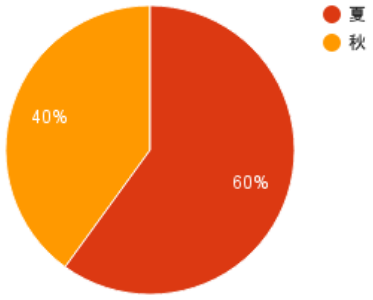

図 14　卒研システムで診断したときのフォーシーズンタイプ

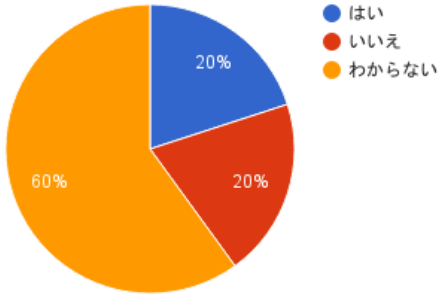

図 15　卒研システムで診断したときのフォーシーズンタイプが自分に当てはまると思うか

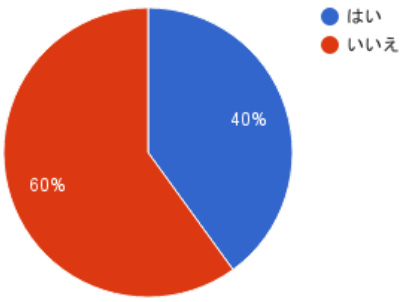

図 16　卒研システムで診断したときと従来のシステムで診断した結果が同じであったか

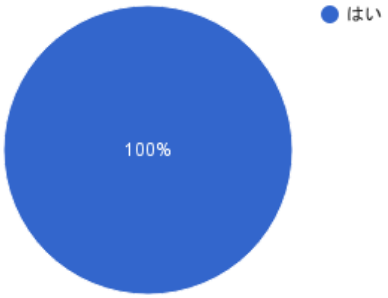

図 17　従来の診断方法より卒研システムで診断したほうが容易か

8.2　考察

アンケート調査を実施した結果、サマータイプとオータムタイプだけに偏っていることがわかる。これは光の当たりが弱く彩度が低いためだと考えられる。

卒研システムで診断した結果が自分に当てはまるかという項目では、大半がわからないという結果になった。ヒアリングをしてみると、はいかいいえで答えた人は以前にパーソナルカラー診断を受けたことがある、普段から自分に似合う色かどうかを意識していることがわかった。わからないと答えた人は、パーソナルカラーという単語を初めて聞き、普段のファッション選びにおいて似合うかどうか自分ではわからないというタイプであることがわかった。

卒研システムの結果の信頼性については、保証することができなかった。卒研システムと従来のシステムでの診断結果が同じ人が少ないからである。また、診断結果の色について似合うかわからないという結果が多数だったからである。

本研究の目的である、手軽に診断できるかどうかについては、達成することができた。

8.3 今後の課題

今回の診断で使用する写真は、撮り方に条件を付けただけだった。だが、写真とはカメラの機種や光源、天気によって色味が変わってしまう。また、診断サイトでのユーザの手間と時間を減らすことを目的にしていたので、動作は写真を選択するだけにしたかったのだが、各パーツを選ぶという動作が必要になってしまった。これはOpencv でパーツの検出がうまく行かなかったからである。

それを踏まえた今後の課題は以下のとおりである。

8.3.1 写真の自動補正機能を追加

写真を撮るとき、光源によって写真全体の色味が変わる問題あった。これを演色性という。白っぽい光を当てると対象物は青みが増し、赤っぽい光を当てると対象物は黄みが増すため、パーソナルカラーの診断結果に大きく影響を与えてしまう。色は自然界から取っているので、自然界と同じく太陽光下で撮影しても、天気によって色味がかわってしまう。

ユーザに修正してもらうことは負担がかかりすぎるため、プログラム側で何らかの処理が必要になる。例えば絶対に色が同じものを写真に写り込ませて、それを基準に色味を合わせることを求められる。

8.3.2 ブラウザのキャッシュ問題

Web アプリケーションを利用すると、写真を送信あとにパーツ画像が表示される。そのときに使いたいパーツ画像が無ければ写真を送り直す仕様となっているが、画像のキャッシュが残っていると新しく送信した画像のパーツ画像が表示されず、前に送信した画像のパーツが表示される。ブラウザのキャッシュを毎回消すかブラウザをシークレットモードで利用することでこの問題を回避することができるが、再び画像を送信したときは画像の保存名を変えるなどして、システム側で対策をすることが望ましい。

8.3.3 パーツ検出の精度向上

Opencv でパーツを検出させると、口が複数検出されるなど精度に問題があった。そのため、ユーザにパーツを選んでもらうという手間が発生した。今回は Haar–Like 分類器を使っているため、顔に光のむらがあることが上手く行かない原因である。今後は、検出時の値を調整するか、違う分類器でパーツ検出をすることが望ましい。

8.3.4 似合うかどうかについて

アンケートを実施した際、多数が結果ででたパーソナルカラーが似合うかどうかわからないとい解答した。これは、似合う色と似合わない色を身に着けた際の変化を、実感できなかったからだと考える。そのため、Web 上でも似合う色と似合わない色を身に着けたときのシミュレーションができるとよい。

8.3.5 似合うファッションついて

パーソナルカラーとはその人に似合う色のことを指すのだが、ファッションについては、似合う色のものを身にまとっても必ずその人に似合うわけではない。骨格やその人がもつ雰囲気に左右されてしまうからである。そのため、服のアドバイスをするときはパーソナルカラーだけでなく、骨格の知識やその人がどのような人なのかを知ることが必要不可欠となる。

9　結論

本研究ではパーソナルカラーの自動検出をする Web アプリケーションの開発を行った。動作しているのは自動判別をするために OpenCV で目、鼻、口の検出を行い、各パーツの HSV 値で似合う色を判別するシステムである。

その結果、1 枚の写真から OpenCV で目、鼻、口を切り取り、各パーツの HSV 値の平均を算出することができた。HSV 値を出せたことで、フォーシーズンタイプに分類することができ、パーソナルカラーがわかるようになった。ただし診断結果は信頼性があるとはいえないため、さらなる信頼性の確証と正確性を上げることが求められる。今後、個人のファッションの参考にできるだけでなく、アパレル業界とヘアメイク業界に貢献することが期待できる。

謝辞

本研究を進めていく上で、担当教員である大垣斉准教授に御指導及び御協力をいただきました。また、情報教育システム研究室のメンバー及び卒業生の方々、友人の皆様方にアンケート調査、ご助言を賜りました。ここに深く感謝の意を表します。

参考文献

[1] イエローベースとブルーベース. http://rock77.fc2web.com/main/color-topix/p-color.html, Oct. 2017.

[2] パーソナルカラー. http://rock77.fc2web.com/main/color-topix/p-color.html, Oct. 2017.

[3] 森本のり子. 骨格診断とパーソナルカラー診断で見つける似合う服の法則. http://p-brand.com/, Jun. 2016.

[4] opencv. http://opencv.org/, Des. 2017.

[5] flask. http://flask.pocoo.org/, Des. 2017.

[6] irouse. http://www.geocities.jp/net_t3/color/personal04.html, Des. 2017.

付録 A　ソースコード

A.1　パーツ検出

Listing 1　顔写真から目、鼻、口を切り取るプログラム

```
#coding: utf-8

import os
import cv2
import numpy

def cut_img(img):
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    eye_cascade = cv2.CascadeClassifier('haarcascade_mcs_righteye.xml')
    nose_cascade = cv2.CascadeClassifier('haarcascade_mcs_nose.xml')
    mouth_cascade = cv2.CascadeClassifier('haarcascade_mcs_mouth.xml')

    current_dir = os.path.abspath(os.path.dirname(__file__))
    img.save(os.path.join(current_dir, img.filename))
    filename = img.filename

    img = cv2.imread(os.path.join(current_dir, img.filename), cv2.IMREAD_COLOR)
    os.remove(os.path.join(current_dir, filename))

    if(img is None):
        return False

    else:
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        face = face_cascade.detectMultiScale(gray)

        if type(face) is tuple:
            return False

        i=0

        for (x,y,w,h) in face:
            roi_gray = gray[y:y+h, x:x+w]
            roi_face = img[y:y+h, x:x+w]
```

```
        eyes = eye_cascade.detectMultiScale(roi_gray, 1.3, 5 )
        nose = nose_cascade.detectMultiScale(roi_gray, 1.3, 5 )
        mouth = mouth_cascade.detectMultiScale(roi_gray, 1.3, 5 )


    if type(eyes) is tuple:
        roi_eyes = False
    else:
        for (ex,ey,ew,eh) in eyes:
            i+=1
            cv2.imwrite(current_dir+'/static/image/'+str(i)+'.jpg', roi_face[ey:ey+eh,

    if type(nose) is tuple:
        roi_nose = False
    else:
        for (nx,ny,nw,nh) in nose:
            i+=1
            cv2.imwrite(current_dir+'static/image/'+str(i)+'.jpg', roi_face[ny:ny+nh,

    if type(mouth) is tuple:
        roi_mouth = False
    else:
        for (mx,my,mw,mh) in mouth:
            i+=1
            cv2.imwrite(current_dir+'static/image/'+str(i)+'.jpg', roi_face[my:my+mh,

#import pdb; pdb.set_trace()
```

A.2 パーソナルカラー判別

Listing 2 目、鼻、口の HSV をもとにフォーシーズンを判定するプログラム

```
#coding: utf-8

import cv2
import numpy as np
import os
import glob

def personal_color(eye,nose,mouth):
    current_dir = os.path.dirname(__file__)
```

```
if eye == 0:
    roi_eyes = False
else:
    roi_eyes = cv2.imread(current_dir+'/static/image/'+str(eye)+'.jpg', cv2.IMREAD_COL

if nose == 0:
    roi_nose = False
else:
    roi_nose = cv2.imread(current_dir+'/static/image/'+str(nose)+'.jpg', cv2.IMREAD_CO

if mouth == 0:
    roi_mouth = False
else:
    roi_mouth = cv2.imread(current_dir+'/static/image/'+str(mouth)+'.jpg', cv2.IMREAD_



def season(roi_parts):
    hue = 0
    saturation = 0
    value = 0

    if roi_parts is False:
        no_parts = roi_parts


    hsvcat = cv2.cvtColor(roi_parts, cv2.COLOR_BGR2HSV)

    tyuusin = (hsvcat[hsvcat.shape[0]/2, hsvcat.shape[1]/2]),\
          (hsvcat[hsvcat.shape[0]/2+1, hsvcat.shape[1]/2+1]),\
          (hsvcat[hsvcat.shape[0]/2+1, hsvcat.shape[1]/2-1]),\
          (hsvcat[hsvcat.shape[0]/2-1, hsvcat.shape[1]/2+1]),\
          (hsvcat[hsvcat.shape[0]/2-1, hsvcat.shape[1]/2-1])

    for h,s,v in tyuusin:
        hue += h.astype(np.float64)
        saturation += s.astype(np.float64)
        value += v.astype(np.float64)

        ave_hue = hue/5
```

```
        ave_saturation = saturation/5
        ave_value = value/5


    spring = 0
    summer = 0
    autumn = 0
    winter = 0

    if ave_saturation > (255/2) and ave_value > (255/2):
        spring += 1
    elif ave_saturation < (255/2) and ave_value < (255/2):
        autumn += 1
    elif ave_saturation < (255/2) and ave_value > (255/2):
        summer += 1
    elif ave_saturation > (255/2) and ave_value < (255/2):
        winter += 1

    return spring, autumn, summer, winter


if type(roi_eyes) != bool:
    spring, autumn, summer, winter = season(roi_eyes)
else:
    print('no')

if type(roi_nose) != bool:
    spring, autumn, summer, winter = season(roi_nose)
else:
    print('no')

if type(roi_mouth) != bool:
    spring, autumn, summer, winter = season(roi_mouth)
else:
    print('no')



list = [spring, summer, autumn, winter]
re_season = max(list)
```

```
    if re_season == spring:
        return 1
    elif re_season == summer:
        return 2
    elif re_season == autumn:
        return 3
    elif re_season == winter:
        return 4


    os.remove(current_dir+'static/image/*.jpg')
```

A.3 Web アプリケーションとしての動作部分

Listing 3 Web で受け取った情報をプログラムに渡している

```
# coding:utf-8
import cv2
import numpy
from glob import glob
import os
from flask import Flask, render_template, request
from cut_parts import cut_img
from color import personal_color
from os.path import join, relpath

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html', message='Select the face image.')

@app.route('/uploader', methods=['POST'])
def upload_file():
    f = request.files['the_file']
    if cut_img(f) != False:
        cut_img(f)

#       import pdb; pdb.set_trace()
        path = os.path.join(os.path.abspath(os.path.dirname(__file__)), "static/image")
        image_names = [relpath(img_path, path) for img_path in glob(join(path, '*.jpg'))]
```

```
        my_liist = []

        for image in image_names:
            my_dic = {}
            my_dic['image_name'] = '/static/image/' + image
            my_liist.append(my_dic)

        return render_template('cut.html', message='Select the eye,nose and mouth. if no yo
    else:
        return render_template('index.html', message='Re-select the another face image.')

@app.route('/uploader/select', methods=['POST'])
def select_parts():
    if request.method == 'POST':
        eye = int(request.form['number1'])
        nose = int(request.form['number2'])
        mouth = int(request.form['number3'])
        four_seasons = personal_color(eye, nose, mouth)

    if four_seasons == 1:
        return render_template('season_temp.html', base_color='yellow', season='spring')
    elif four_seasons == 2:
        return render_template('season_temp.html', base_color='blue', season='summer')
    elif four_seasons == 3:
        return render_template('season_temp.html', base_color='yellow', season='autumn')
    elif four_seasons == 4:
        return render_template('season_temp.html', base_color='blue', season='winter')

@app.route('/uploader/select/parsonalcolor', methods=['POST'])
def parsonal_color():
    return render_template('index.html', message='Select the face image.')

if __name__ == "__main__":
    app.run()
```