19 不適切画像自動フィルタリングシステムの作成

13 番　田中拓人

指導教員　菅野　研一

1. はじめに

図形処理工学で画像処理を学び，モザイク処理など様々なことができることを知り，興味を持った.画像処理技術を用いたソフトを作ってみたいと思った.

2. 目的

インターネット上には青少年にふさわしくない画像が多数流通している.

画像は文字によるフィルタリングのように簡単に処理できない．自動的にフィルタリングできれば青少年を有害な情報から守ることができる.

そのために，画像処理関連のライブラリが豊富な OpenCV を用いてフィルタリングソフトを作成する.また，OpenCV の言語は Java を用いてさらに理解を深める

3. 開発環境

表 1 開発環境

OS	Windows 10
IDE	eclipse
使用言語	Java
ライブラリ	OpenCV

3.1 OpenCV

OpenCV とはインテルが開発公開したオープンソースのライブラリである．画像処理・画像解析および機械学習の機能を持つため，そういった分野での活用が活発である.また，対応している言語が C/C++，Java，Python，MATLAB と幅広く対応しており,プラットフォームも Unix 系 OS,Linux,Windows,Android,iOS に対応している.

3.2 Swing

Swing は,Java の GUI ツールキットである.Swing は AWT よりも洗練された GUI コンポーネントを提供するために開発された.

4. 研究概要

4.1 処理の手法

PC の画面上に映っているものをリアルタイムで顔認識プログラムで人かどうかを判断させる.顔の領域から色を抽出し画像全体に占める割合を求める．その割合が大きいと衣類がない状態と判断できる．衣類がない状態であればモザイクをかける.

図 1 の場合,顔領域の色と類似している色が画像に内に多いため衣類がないと判断される．図 2 の場合顔領域の色が画像内に少ないため衣類があると判断される.

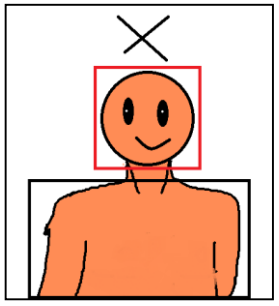

図 1. 衣類なし

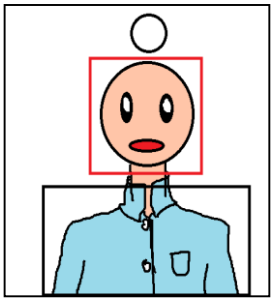

図 2. 衣類あり

4.2　顔検出について

物体の検出器を作成するには,まずは「機械学習」を行う. 機械学習では,学習したい物体の特徴を抽出して、抽出した「特徴量」を機械が学習し,学習データをまとめる.

この機械学習のまとまったデータを「カスケード分類器」と呼ぶ.

本研究では Haar-Like 特徴を用いて顔検出を行う.

4.3　Haar-Like 特徴

黒い四角形の領域に含まれる画素値の総和から白い四角形の領域に含まれる画素値の総和を引いた値がこの特徴量になる.

図 1 の中央の画像は目の辺りの領域は鼻や頬の領域に比べて暗いという性質を捉えいる.右の画像は目は眉間より暗いという性質を捉えている.

このような特徴をもとに顔と判断している.

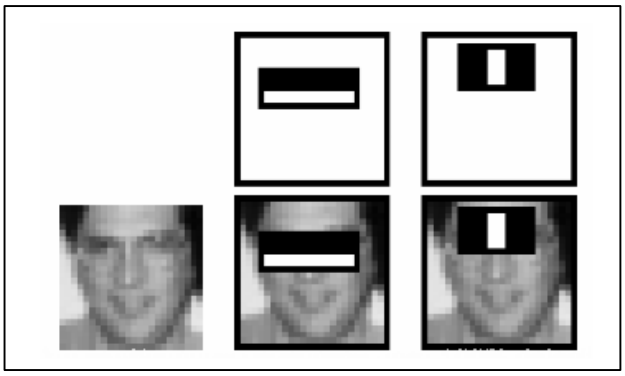

図 3

4.4　HSV 色空間

HSV 色空間とは,色を【色相(Hue)】【彩度(Saturation)】【明度(Value・Brightness)】の 3 要素で表現する方式.

肌検出を RGB を用いて行おうとすると,暗い場所では全体の RGB は低く,明るい場所では全体の RGB は高くなってしまうため,閾値の範囲が広くなってしまい肌色以外の色も検出されてしまう可能性が高くなる.

HSV では色相だけを用いて肌検出ができるため本研究では HSV を使う.

4.5　顔領域から色の抽出

検出した枠線で囲んでいる顔の領域から目と口部分を取り除き,平均値を求める.

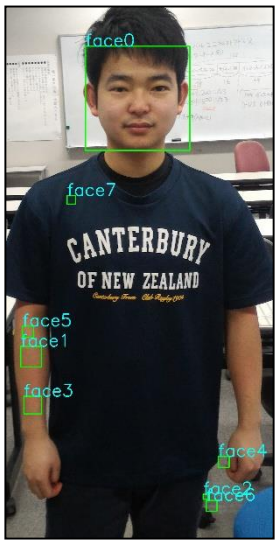

図 4 検出された顔に枠線を描画

```
Start.
Read testko.jpg.
Detected 8 faces.
Detected 3 eyes.
Detected 24 mouths.
face0::(RED:132 GREEN:106 BLUE83)
face0::(H:28 S:37 V:51)

face1::(値なし)

face2::(値なし)

face3::(RED:141 GREEN:119 BLUE69)
face3::(H:41 S:51 V:55)

face4::(値なし)

face5::(値なし)

face6::(値なし)

face7::(値なし)
```

図 5. 検知された肌の平均値

4.6　RGB を HSV に変換

RGB のうち,最大値を MAX,最小値を MIN とする.

【色相 Hue】

$$R = MAX \quad H = 60 \times \frac{(G-B)}{(MAX-MIN)} \qquad 式\ (1)$$

$$G = MAX \quad H = 60 \times \frac{(B-R)}{(MAX-MIN)} + 120 \qquad 式\ (2)$$

$$B = MAX \quad H = 60 \times \frac{(R-G)}{(MAX-MIN)} + 240 \qquad 式\ (3)$$

【彩度 Saturation】

$$S = \frac{(MAX-MIN)}{MAX} \qquad 式\ (4)$$

【明度 Value】

$$V = MAX \qquad 式\ (5)$$

図 4.2 が H の平均値だけに閾値を設定した場合である.

$$H - 15 \leq H \leq H + 15$$

肌色をしっかりと検知しているが,背景など肌以外の場所も検知してしまっている.

図 4.3 は HSV の平均値に閾値を設定した場合である.

$$H - 15 \leq H \leq H + 15$$
$$S - 25 \leq S \leq S + 25$$
$$V - 25 \leq V \leq V + 25$$

背景や服の文字など誤検知が大分なくなった.

本研究では以降この閾値に設定して進める.

図 6.1

図 6.2

4.7　肌色領域の割合からモザイク処理をかける

画像全体に占める肌色の数が顔領域の肌色の数の割合でモザイクをかける.

モザイクをかける肌色の割合は,サンプル画像 20 点に対して処理を試みたところ,最も良い結果が得られた 6 倍とした.

モザイク処理は画像を 1 度縮小してから拡大するという処理を行いモザイクをかけている.

図 6.1 が元画像で,図 6.2 がモザイク処理をした後の画像である.顔領域の肌色の数が 11449 で画像全体に占める肌色の数が 108717 で,全体に占める割合が顔領域の 6 倍以上あるのでモザイクがかかっている.

図 7.1

図 7.2

```
Start.
Read DSC_0226.png.
Detected 2 faces.
Detected 0 eyes.
Detected 9 mouths.
face0::(RED:104 GREEN:62 BLUE51)
face0::(H:12 S:50 V:40)

face1::(値なし)

顔領域の肌色数:face0 11449
画像全体の肌色数face0 108717

done.
```

図 8

4.8　swing での表示

「画像選択」ボタンで画像を選択する.「表示」ボタンで画像が表示される.画像全体の肌色数が少なければ元画像が表示される.「保存」ボタン表示されている画像を任意の場所に保存できる.

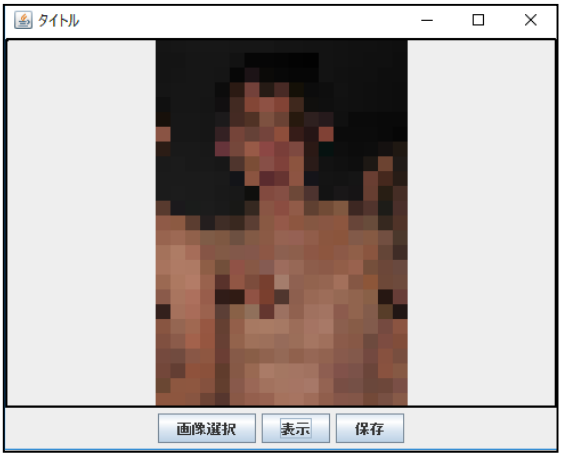

図 9

5. 終わりに

不適切画像フィルタリングシステムとして画像のみに対応したフィルタリングシステムを作成することができた. 現在 351×469 ピクセルの画像での処理に 15 秒を擁しており,リアルタイム処理や動画への適用には至っていない. これからは今後の課題としたい.

6. 参考サイト

[1] OpenCV modeles, https://docs.opencv.org/3.1.0/

[2] OpenCV でモザイク処理, https://qiita.com/ciela/items/f29fb91e28360b514307

[3] OpenCV で顔認識など特徴ある領域の検出, https://qiita.com/yokobonbon/items/f7ff24cc449a1fe6ba4b

[4] Swing で Mat 型画像を GUI 上に表示, https://qiita.com/JackMasaki/items/79b883ca5084d7586008