6　Java Script を用いた画像編集 Web アプリケーション

4 番　遠藤　公志朗

指導教員　飯坂　覚

1.　はじめに

最近、若者の間で SNS などに編集された画像を投稿している人が多く見受けられ、PC やスマートフォンにインストールされた画像編集アプリが使われていることがある。私はこの画像処理の仕組みを理解し、JavaScript を用いた画像編集に必要なライブラリを導入した Web アプリケーション(以下、Web アプリ)を作成したので報告する。

2.　研究概要

2.1　Web アプリ

サーバ上に画像編集アプリを実装するために、外部からアクセスが可能な画像編集 Web アプリ(http://www.iwate-stars.jp/~koshiro/sotuken/edit/index.html)を作成した。

この Web サイトでは、編集したい画像をユーザーがアップロードすることで Canvas Ⓐに画像が表示される。その後 Web 上に用意されている画像編集ボタンを押すことで入力画像の下に編集機能が反映された画像が Canvas Ⓑが表示される形で実装した。Canvas を使うことで、ブラウザに対応した画像を描画することができる。

Ⓐ及びⒷの画像はデフォルトで PNG 形式の画像が保存される。

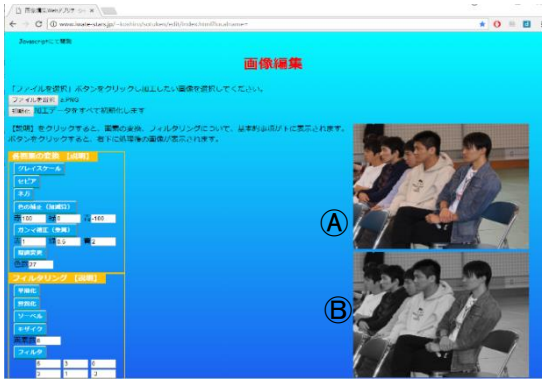

図 1　完成した画像編集 Web アプリ

2.2　開発環境

開発環境を表 1 に示す。GoogleChrome、Web アプリを実装するので HTML5 と画像処理に必要な JavaScript、そして処理効率を早いことやより少ない記述で書けるようにする為に jQuery ライブラリを導入し、OS は Windows 10 で行った。

表 1　開発環境

OS	Windows 10
ブラウザ	Google Chrome(ver.64.0.3282.140)
使用言語	JavaScript　HTML5
ライブラリ	jQuery

3.　サイトの構成

HTML5 では、File API を利用して、選択されたローカルファイル(ここでは入力画像)とやり取りする方法がある。ここで jQuery を使うことで処理を効率よく行い、CanvasⒶに入力画像を表示させている。

Web 上に用意されているボタンを押すことで onClick イベントを呼び起こしている。

onClick イベントとは、ドキュメント内の要素をクリックした際に起こるイベント処理のことであり、ここでクリック後に処理してほしい関数を指定することができる。

画像処理をさせたい場合、この onClick イベントを毎回呼び起こすことにより、入力画像が表示されている CanvasⒶの画像情報を送り、処理している。ユーザーが任意の補正数値等を入力し処理させたい場合にはそれらの数値も送っている。

処理された画像は CanvasⒷに表示され、完了となる。

上書き保存機能のみ、CanvasⒷの画像を CanvasⒶに置き換える処理を行っている。

4.　画像編集

今回開発した画像編集機能は全体で 13 種類あり、大きく分けると画素の変換処理とフィルタリングに大別される。ここではそれぞれの処理の違いについて説明する。

4.1　各画素の変換

画像は画素(ピクセル)の集合体である。各画素は赤・緑・青の 3 原色が 0~255 の段階で表現されており、入力画像の画素の縦位置と横位置、色情報を処理することで出力画像を変化させている。

各画素の変換にてできる処理は次の通りとなっている。

図 2　各画素の変換の編集ボタンと数値入力

①　グレイスケール

画像の RGB を取得し、3 原色の値が同じであるとき、その色が彩度 0 のグレーになる。

出力画素= (0.2126*赤+0.7152*緑+0.0722*青)　(1)

②　セピア

グレイスケールの処理をした後、3 原色に強弱をつけることでセピアにする。ここで、Y はグレイスケール変換処理をした後の画素値である。

$$\begin{cases} 赤 = (Y/255)*240 \\ 緑 = (Y/255)*200 \\ 青 = (Y/255)*145 \end{cases} \qquad (2)$$

③　ネガ

3 原色の色を最大値(255)から画素値を引き、反転することでネガ変換(写真のフィルム)にする。

④　色の補正

3 原色の強さは 0~255 の段階で表されており、ユーザーが 3 原色に好きな値を入力することで色の強さを変更することができ、-255~255 まで値を変化させることができる。

⑤　ガンマ補正

ディスプレイは、RGB の信号をそのまま画面に表示できるわけではない。

RGB の強さをそのままディスプレイに渡すと、画面が暗めになり RGB のバランスが崩れてしまうので色も正しく表示できない。

これを避けるために、RGB の値を予め大きくし、ディスプレイの特性をできるだけ直線に近づけることをγ(ガンマ)補正という。

ユーザーが好きな値を入力することで任意のガンマ補正をすることができる。

値を 0<γ<1 にした場合色は明るく(濃い)色となり、1<γ の場合暗く(薄く)なる。

⑥　階調変更

フルカラー(16 万色)を少ない色にする。色数を 8 とすると、一つの原色は 2 段階になる。

色の強度段階は 0~255 だから、128 未満ならば 0、128 以上ならば 255 とする 2 段階にする。

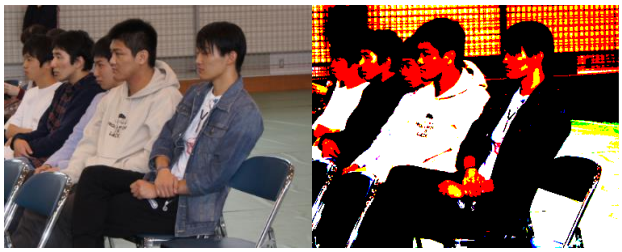

図 3　元画像と階調変更(色数 8)をした様子

4.2　フィルタリング

フィルタリングとは、入力画像の注目する画素値だけでなく、その近傍(周囲)にある画素値も利用し、出力画像の画素値を計算する処理のことである。デフォルトで平滑化、鮮鋭化、ソーベルの３つの機能が用意されているが、ここでは、ユーザーが好きな値を入力することで任意のフィルタ

図 4　フィルタリングの編集ボタンと数値入力

を作成できるようにした。フィルタリングにてできる処理は次の通りとなっている。

① 平滑化

画像は基本的に隣同士の画素は似た画素値を持つ特徴がある。

各画素の色を、周囲の画素の平均化させる処理をすることでノイズを除去し、色の変化が滑らかになり、出力画像がぼやけるようになる。次のような畳み込み処理を施した。

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

② 鮮鋭化

鮮鋭化はエッジ部分を強調(色の変化を強調)するフィルタのことであり、エッジとは，画像中の明るさ（濃淡）あるいは色が急に変化している箇所を指している。次のような畳み込み処理を施した。

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

③ ソーベル

画像の濃度差を利用して画像の一部を浮きだたせるフィルタのことである。

平滑化フィルタをかける際に「注目画素との距離に応じて重み付けを変化させた」ものがソーベルフィルタという。

ここでは、入力画像にグレイスケール変換を施した後、フィルタ C をかけ、その後フィルタ D をかける畳み込み処理を施した。

$$C = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

④ 任意のフィルタの作成

好きな値を入力することで任意のフィルタを作成できる。各要素の値は−5～5 程度が適切となっている。代表的なフィルタの例は以下のようになっており、動作することが確認されている。

ガウシアン：より細かい平滑化

$$\begin{bmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{bmatrix}$$

鮮鋭化：斜め隣接画素の影響も考慮されたもの

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & -9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

ラプラシアン：鮮鋭化、エッジをまとめたもの

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Prewitt：エッジ検出（縦方向）

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Sobel：エッジ検出（横方向）

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

エンボス：エッジ検出の一つ

$$\begin{bmatrix} 5 & 3 & 0 \\ 3 & 1 & -3 \\ 0 & -3 & -5 \end{bmatrix}$$

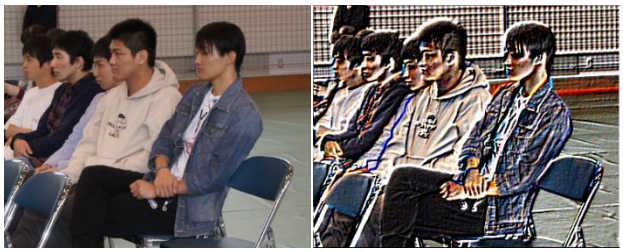
図 5　元画像とエンボスフィルタを使用した様子

4.3　その他

入力画像の画素に関係ない画像処理が以下のものになる。編集した画像の一時的保存は、処理を重ねがけしたい時に用いるように作成した。

①　グラデーション

ここでいうグラデーションとは、入力画像の左端から右端に向かって、赤を弱くし青を強くしており、色彩を段階的に変化させることを指している。ここでは左から右にかけて、次第に赤を弱くし、青を強くしている。

②　透明化

画素の透明度を上げて背景の画像を透かして見えるようにしている。中心部分を元画像のままにし、周囲になるにつれて透明度を上げていく処理になっている。

③　編集した画像の一時的保存

出力画像を入力画像に変換することで上書き保存のような機能を付け加えた。これにより各操作を連続的に行い、重ねがけができるようになっている。例えば、ネガ→上書き保存→ネガとすると上画面にグレイスケールされた画像が表示され、それが元の画像となる。

5.　考察

本研究を進めていく中で、HTML5(Canvas)、JavaScript 向けの画像処理ライブラリが Web 上で数多く紹介されていた。

これらを見ると基本的な処理の内容は、同じものが多いが、画像の情報の受け渡しの方や HTML5 の機能のみで処理をする方法や外部ファイルにすべて処理してもらう方法、C#や C++を使って処理する方法などの違いがあることに気が付いた。

様々な方法がある中、どの方法が目的の達成に近づけるかを吟味するのが重要なのではないかと私は感じた。

6.　終わりに

JavaScript を用いてブラウザ上で画像編集ができる Web アプリを作成した。本研究を通して画像処理に対する理解は大変深まったと感じる。

JavaScript や HTML5 を今回用いて研究をしていたが、Canvas から画像の情報をどのように渡すか、処理がうまくいかない場合、どこでエラーが起きており対処すればいいのかがわかりにくかったため、苦労した。解決策として Canvas のサイズを固定化させたことや jQuery を用いて処理を簡略化させるなど様々な工夫をして何とか乗り越えることができた。

研究を進めるために様々なサイトや文献を見ると、多様なプログラムの書き方があると改めて感じた。恐らく私が知らないだけで、より簡単に画像処理ができるかもしれない。しかし、画像処理の仕組みや Web アプリの作成する目標が達成できたことが大きいと思う。

今回の研究で JavaScript と HTML5 の理解が深められた。JavaScript は書き方の変化が非常に激しい言語なので対応できるほどの技術を今後も身に着けたいと思う。

7.　参考文献

[HTML5] Canvas の表示内容にフィルターをかける（グレースケース、ネガポジ反転） -YoheiM . NET.html
URL:http://www.yoheim.net/blog.php?q=20120907

ローカル画像ファイルの CANVAS 表示＜File API、CANVAS＜Javascript＜木暮.html
URL:http://www.kogures.com/hitoshi/javascript/Canvas/image-local.html