

# 情報システム工学演習 II 画像処理

## 資料スキャナー

08D20091 Do Thai Don

2022年12月20日

### 1 背景

デジタル化の流れで、紙媒体である文書や資料などを電子的に格納したり管理したりすることが流行っている。大学や職場においてオンライン授業やテレワークが普及しつつ、資料を電子ファイルに変換して扱うことが多くなる。そこで、専用スキャナーの他には、スマホのカメラを利用して本や資料の画像を自動的に正体化するアプリがますます使われるようになっていく。本演習では、これらのアプリにおける画像処理を想定し、撮影画像から四角い本や資料を検出してみる。

### 2 実装したツール

本演習で実装したツールは、入力画像に対して、コントラストと輝度、さらに画像にかけるフィルタの設定値・引数を調整できる。また、それらの設定を調整しながら、画像中の四角形のものを自動的に検出することができ、エッジ検出により本や資料を発見することを臨む。さらに、発見したものを枠に囲めそれを正体化して出力する。これは、演習課題の中級編「画像の正対化」と同様的な目的であるが、画像の四隅をクリックするではなくエッジ検出のアプローチを取って、画像処理におけるフィルタの動作を学ぶことを目的とする。

#### 2.1 外部仕様

ユーザは、本や資料をカメラで撮った画像を input フォルダーに用意する。jupyter notebook 上のツールを起動する際、画像ファイル名を事前に入れておく。

ツール実行は、notebook におけるコードブロックを cmd+enter コマンドで順番に実行する。そこで、ツール本番のコードが起動され、ツールの UI が画面に反映される。

ツール画面では、トラックバーにある設定値をマウスで変更する。ツールを終了するには、「s」・「q」・「esc」のいずれかのキーを押す。ここで、「s」キーを押した場合、終了直前の設定値と四隅の座標が保存される。ツール画面が終了した後、1つの四隅が発見された場合、正体化

した画像を出力する。

#### 2.2 内部仕様

本演習で実装するツールは、各フィルタの動作を学ぶ目的をとるため、できるだけフィルタ実装における引数を変更し効果を試すことができるようにする。ここで、6 本のトラックバー (Contrast, Brightness, Blur kernel size, Blur sigmaX, Canny thres1, Canny thres2) を設ける。

- Contrast と Brightness は、入力画像の明暗の差と輝度の変更に用いられる。
- Blur kernel size と Blur sigmaX は、Gaussian ぼかしフィルタにおけるカーネルサイズと標準偏差の引数として用いられる。
- Canny thres1 と Canny thres2 は、Canny エッジ強調フィルタに用いられる。

処理手順として、まず、入力画像のサイズを減らす。これは、後からの画像処理を早くするためである。ツールの画面が起動したら、上記のトラックバーから読み取ったそれぞれの値に対して、画像が次の順に変換される。

1. 画像のコントラストと輝度を、物体とバックグラウンドが区別できるように変更する。
2. カラー画像をグレイスケールに変換。
3. グレイスケールの画像をぼかす。
4. ぼかした画像をエッジ強調フィルタにかける。
5. 上の処理で、本や資料の辺・エッジが綺麗に抽出されないことが発生するので、辺にある白画素間のギャップを埋める処理を入れる。
6. エッジ強調してから、輪郭線 (contour) を検出する。検出したすべての輪郭線は曲線を含むので、適切に直線のようなものを検出し、さらに 4 本の辺を持つ輪郭のうち面積が一番大きなものを我々が求めたい物体の枠にする。

最後に、上の 6 個の処理の結果画像を、画面に反映する。物体枠を綺麗に描けたら、「s」キーを押して、上の処理

で設定したトラックバーの値と検出した四角形の輪郭の頂点を書き下す。これを用いて、最初にサイズ変更をした画像について正体化の処理を行い、結果画像を output フォルダに出力する。

### 2.3 使用した関数やライブラリ・実行環境

本演習では、デフォルト以外のライブラリは使わない。

## 3 実行結果

### 3.1 成功例 1



図 1: 成功例 1 の入出力画像

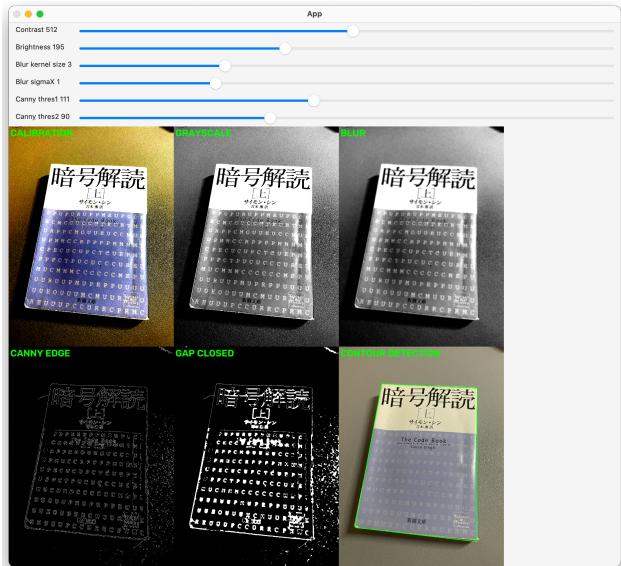


図 2: 成功例 1 のツール画面

図 1 と 2 で示した例は、入力画像中の本カバーをうまく検出できた。ツール画面から、コントラストを 512 に上げ、輝度を 195 に少し下げることにより、本のエッジがより見えるようになった。ここで、コントラスト値 512 は  $512/100 = 5.12$  と、輝度値 195 は  $195 - 255 = -60$  と解釈する。一方で、Gaussian ぼかしフィルタのカーネルサイズを  $3(3 \times 2 + 1 = 7$  と解釈) と標準偏差を 3 にする。これは、普段の設定であり、これらの引数を増加させると次のエッジ強調の効果がうまくいかないことが発生する。さらに、エッジ強調処理では、Canny エッジフィルタのしきい値を本のエッジが見えるまで適当

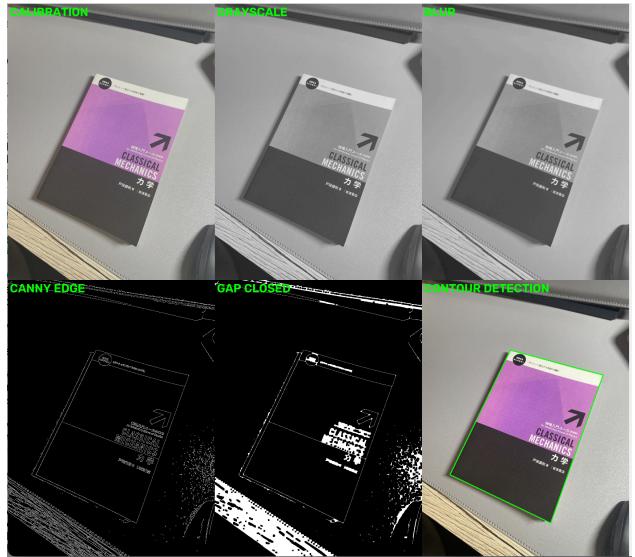


図 3: 成功例 2 のツール画面でとった変換画像



図 4: 成功例 2 の出力画像

に変更して、エッジ検出ができると右下の CONTOUR DETECTION 画像に枠が自動的に描かれる。臨む結果が出たので、「s」を押して図 1b の画像が output され、上記の設定値を settings.txt ファイルに書き込む。

### 3.2 成功例 2

図 3 から、本の下半分の黒い部分をその周りの陰から区別できるようにコントラスト値を減らし輝度を少し上げることによりその後の処理がうまくできたことがわかった。

### 3.3 失敗例

図 5 は失敗例の 1 つである。実装ツールは、四角形の輪郭しか検出しないことにしたので、入力画像中の本や資料の全体をとらないと、四角のエッジを検出することができない。また、この例の本の右辺にブックマークの物体など邪魔なものがつくと、エッジ検出も困難になる。



図 5: 失敗例のツール画面でとった変換画像

[o-finding-books-in-images-using-python-and-opencv/](https://fumio125.github.io/enshu_ip/)

#### 4 考察・感想

本演習の成功例と失敗例から、実装ツールの性能についていろいろ理解できた。

まず、輪郭検出ができるために必須条件として物体全体を画像内に入れるように撮影すること。そうでないと、失敗例のように検出できなくなる。

次に、実装アルゴリズムでは、検出した輪郭のうち面積が一番大きいものを物体として認識することにしたが、今度は、面積のしきい値を設けると1個の物体ではなく複数の四角のものを検出可能という改善点が考えられる。

一方で、複数の成功例を用いてそれらに関する設定値を格納した上で、どの画像にどの設定値が適切であるかというパターンが分析できる。たとえば、ぼかしフィルタにおいては、今回の2つの成功例でカーネルサイズと標準偏差が同様にそれぞれ7と3になっている。また、成功例2の本は白と黒部分にたいして、コントラストを減らし輝度を上がる傾向が見られる。これは、画像の変化量により設定値が決定できるかと考える。

上記の点を踏まえてより工夫すると、おそらく自動で資料をスキャンできるアプリが実現できると想定し、今度の課題にする。

#### 参考文献

- [1] 情報システム工学演習 II 画像処理 演習資料,  
[https://github.com/fumio125/enshu\\_ip](https://github.com/fumio125/enshu_ip).
- [2] Adrian Rosebrock. A guide to finding books in images using Python and OpenCV,  
<https://yasoob.me/2015/03/11/a-guide-to-finding-books-in-images-using-python-and-opencv/>