【7 月】 月間報告書

細胞画像における画像解析

細胞画像から細胞の数をカウントする

JREP 井上朝斐

今回、私は細胞画像から細胞の数をカウントするコードを1から、作ることに挑戦しました。

現段階では一枚ずつ画像入力をしなければなりませんが、夏休み中には多くの画像からの結果をワンクリックで出力できるようにします。

また、核が赤に染色されているものを想定しています。核が染色されていれば、抽出する色を変化させることで対応可能ですのでご申し付けしていただければと思います。

私は細胞を数えることを核の数を数えることと言い換えて、開発しました。

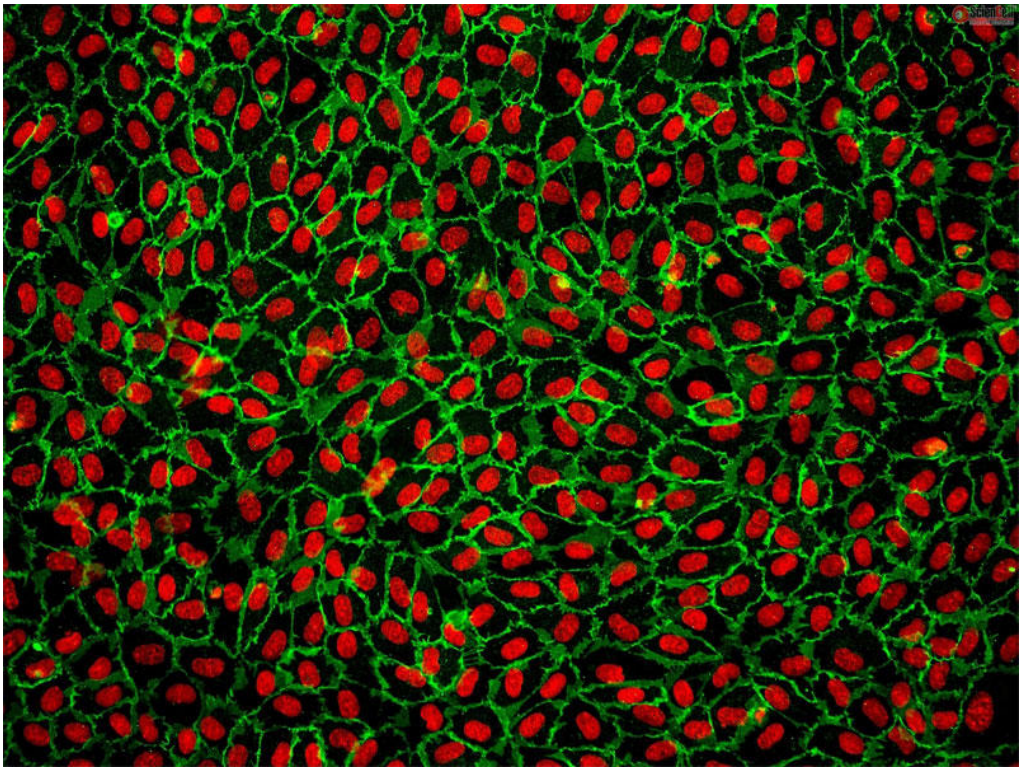
コードは(https://github.com/ash-jp/project_okuda_lab)の cell_count からみることができます。
使い方については別途ドキュメントを書こうと思います。

細胞の数をカウントするフロー

私が、今回考えた手法は以下のようになります。

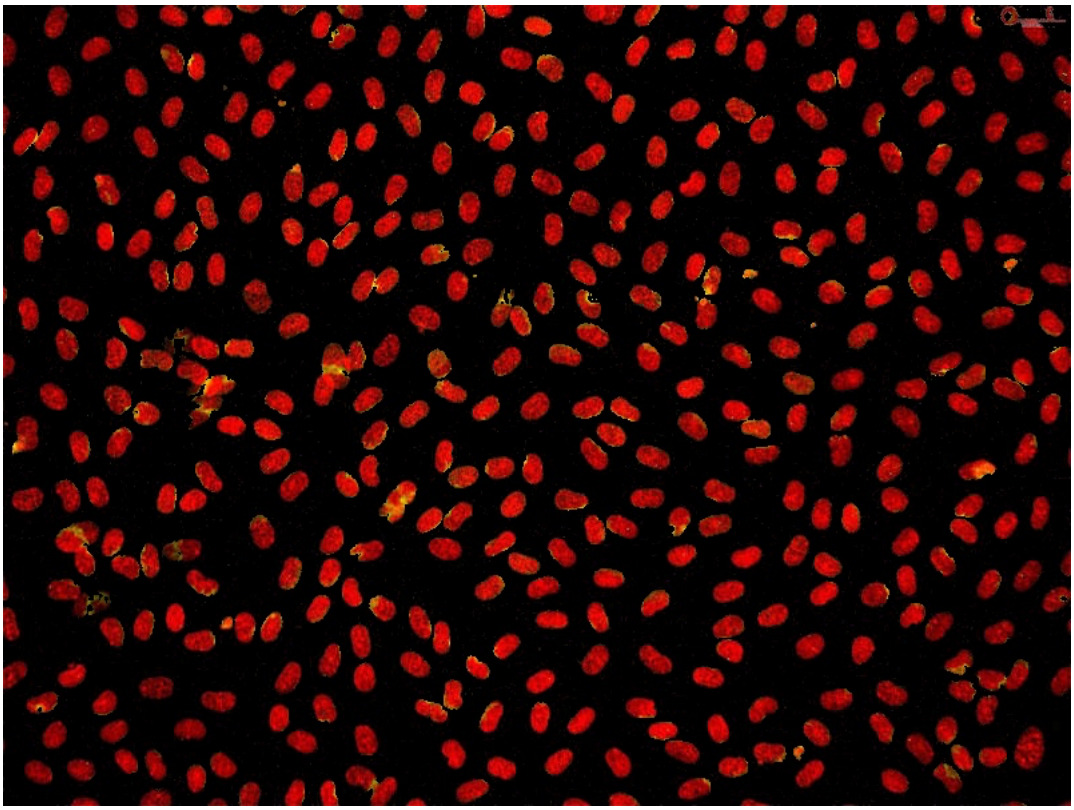
1. 赤色を画像から抽出する。
2. グレースケールに落とし込む。
3. 二値化画像を生成する。(大津の二値化)
4. モルフォジー変換を行う。
5. 数を数える。

今回、は以下の画像を例にして、説明します。



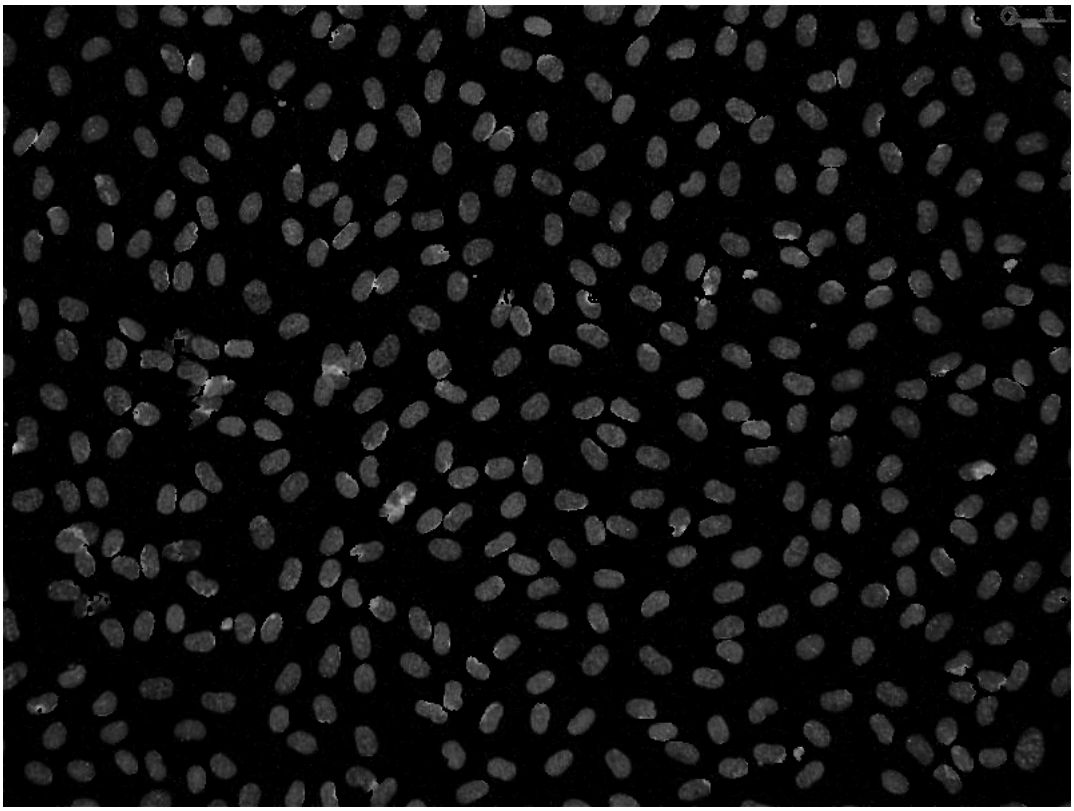
赤色を画像から抽出する。

HSV 色空間と呼ばれる RGB とは異なる（色相、彩色、明度）を基にした色空間で赤色を抽出しました。下図



グレースケールに落とし込む

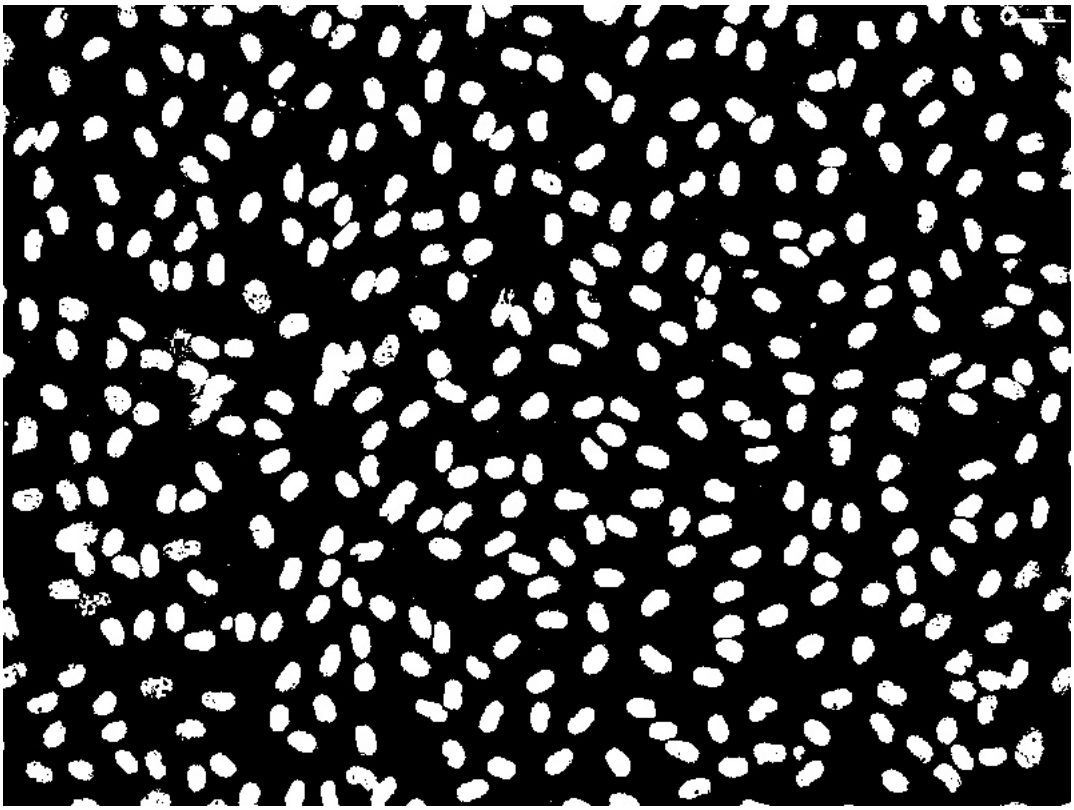
二値化画像を作るためにグレースケールにします。



二値化画像を生成する。（大津の二値化）

二値化画像を作ります。

二値化画像とは閾値を設け、閾値よりも明るいところを白、暗いところを黒と表す手法です。

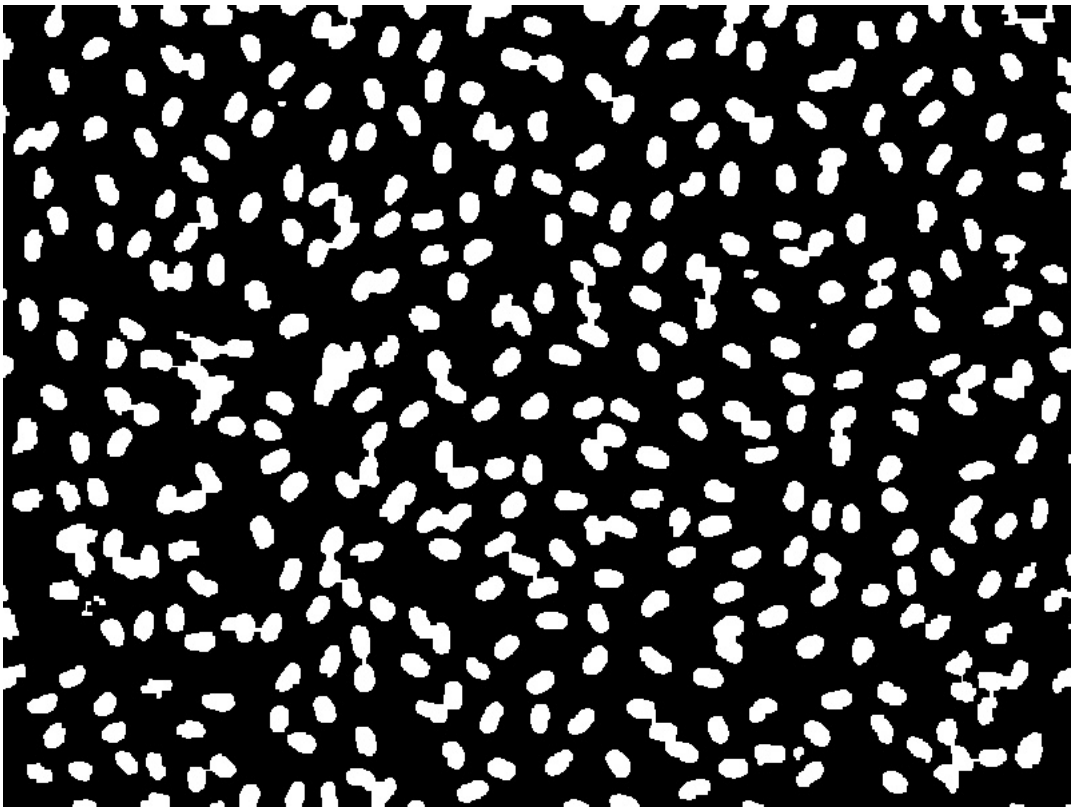


モルフォジー変換を行う

モルフォジー処理を行い、ノイズを除去します。

モルフォジー処理

> *クロージング処理はオープニング処理の逆の処理を指し，**膨張の後に収縮** をする処理です．前景領域中の小さな(黒い)穴を埋めるのに役立ちます*



数を数える。

あとは数を数えるプログラムを書いて終了です。

```
                else:
                    cells += 1
        print('Cells: {}'.format(c
[121]

···   Cells: 329
```

Python コード （jupyter notebok 形式で表示します。）

ライブラリのインポート

In [ ]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import matplotlib
%matplotlib inline
import math
import warnings
warnings.simplefilter('ignore')
```

画像の読み込み

In [ ]:
```
from cv2 import cvtColor
img = cv2.imread("./cell.jpg")
# plt.imshow(cvtColor(img, cv2.COLOR_RGB2BGR))
original = img.copy()
```

赤色を抽出

In [ ]:
```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
# 赤色のHSVの値域1
hsv_min = np.array([0, 64, 0])
hsv_max = np.array([30, 255, 255])
mask1 = cv2.inRange(hsv, hsv_min, hsv_max)

# 赤色のHSVの値域2
hsv_min = np.array([150, 64, 0])
hsv_max = np.array([179, 255, 255])
mask2 = cv2.inRange(hsv, hsv_min, hsv_max)

# 赤色領域のマスク(255:赤色、0:赤色以外)
mask = mask1 + mask2

# マスキング処理
img_red = cv2.bitwise_and(img, img, mask=mask)

# cv2.imwrite("img_r.jpg", img_red)

# plt.imshow(cvtColor(img_red, cv2.COLOR_RGB2BGR))
```

Out[ ]:
```
True
```

グレースケールに

In [ ]:
```
img_gray = cv2.cvtColor(img_red, cv2.COLOR_BGR2GRAY)
cv2.imwrite("img_gray.jpg", img_gray)
# plt.imshow(cvtColor(img_gray, cv2.COLOR_RGB2BGR))
```

Out[ ]:
```
True
```

二値化画像の生成

In [ ]:
```
th, img_thresh = cv2.threshold(img_gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
cv2.imwrite("img_thresh.jpg", img_thresh)
# plt.imshow(cvtColor(img_thresh, cv2.COLOR_RGB2BGR))
```

Out[ ]:
```
True
```

モルフォジー処理

In [ ]:
```
#短形カーネルの生成
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
opening = cv2.morphologyEx(img_thresh, cv2.MORPH_OPEN, kernel, iterations=1)
close = cv2.morphologyEx(opening, cv2.MORPH_CLOSE, kernel, iterations=2)
cv2.imwrite("close.jpg", close)
```

Out[ ]:
```
True
```

細胞のカウント

In [ ]:
```
cnts = cv2.findContours(close, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
```

In [ ]:
```
minimum_area = 20
average_cell_area = 500
connected_cell_area = 1000
cells = 0
for c in cnts:
    area = cv2.contourArea(c)
    if area > minimum_area:
        cv2.drawContours(original, [c], -1, (36,255,12), 2)
        if area > connected_cell_area:
            cells += math.ceil(area / average_cell_area)
        else:
            cells += 1
print('Cells: {}'.format(cells))
```

<感想>

4 月にプログラミングを本格的に始めて、3 ヶ月でこんなことができるようになるとは思いませんでした。やはり、物事をするにはものゴールというものが必要だと思いました。特に画像については授業で全く触れることがなかったので Google Chrome で一生懸命調べ、理解しながらコードを書きました。この程度のことでこんなに時間がかかってしまいましたが、自分の中ではものすごく成長できたと思っています。医療データ的なものを扱うのは（知的に）面白いです。

期末テストがおわってから、この作業を結構やったおかげで最初音階よりも随分と精度がよくなりました。今のままだと、python で少しコードをいじってもらわなければなりません。なので誰でも簡単にこのコードを動かすために Web アプリにして、もっと動的に使いやすく、夏休みの最初はやっていこうと思います。