

Q. 61 - 70

Q.61. 4-連結数

renketsu.pngを4-連結数により、色分けせよ。

4-連結数とは近傍との画素の状態を見る値である。通常、近傍は注目画素x0(x,y)が0でない場合に対して、次のように定義される。

```
x4(x-1,y-1) x3(x,y-1) x2(x+1,y-1)
x5(x-1,y)   x0(x,y)   x1(x+1,y)
x6(x-1,y+1) x7(x,y+1) x8(x+1,y+1)
```

ここで4連結数とは、次式で計算される。

$$S = (x1 - x1 \ x2 \ x3) + (x3 - x3 \ x4 \ x5) + (x5 - x5 \ x6 \ x7) + (x7 - x7 \ x8 \ x1)$$

S = [0,4]の範囲をとり、

- S = 0 は内部点
- S = 1 は端点
- S = 2 は連結点
- S = 3 は分岐点
- S = 4 は交差点 を示す。

入力 (renketsu.png) 出力(answers/answer_61.png)



答え >> [answers/answer_61.py](#)

Q.62. 8-連結数

renketsu.pngを8-連結数により、色分けせよ。

8連結数とは

$$S = (x1 - x1 \ x2 \ x3) + (x3 - x3 \ x4 \ x5) + (x5 - x5 \ x6 \ x7) + (x7 - x7 \ x8 \ x1)$$

において各x¥*の値の0と1を反転させた値を用いる。

入力 (renketsu.png) 出力(answers/answer_62.png)



答え >> [answers/answer_62.py](#)

Q.63. 細線化処理

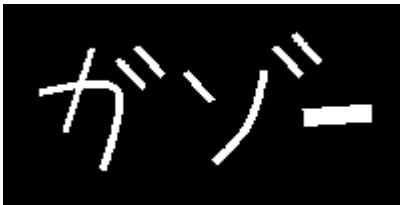
gazo.png を細線化せよ。

細線化とは画素の幅を1にする処理であり、ここでは次のアルゴリズムに沿って処理を行え。

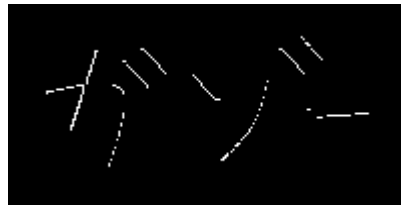
1. 左上からラスタスキャンする。
2. $x_0(x,y)=0$ ならば、処理なし。 $x_0(x,y)=1$ ならば次の3条件を満たす時に $x_0=0$ に変える。(1) 注目画素の4近傍に0が一つ以上存在する (2) x_0 の4-連結数が1である (3) x_0 の8近傍に1が3つ以上存在する
3. 一回のラスタスキャンで2の変更数が0になるまで、ラスタスキャンを繰り返す。

細線化にはヒルディッチのアルゴリズム(Q.64)や、Zhang-Suenのアルゴリズム(Q.65)、田村のアルゴリズムなどが存在する。

入力 (*gazo.png*)



出力(*answers/answer_63.png*)



答え >> [answers/answer_63.py](#)

Q.64. ヒルディッチの細線化

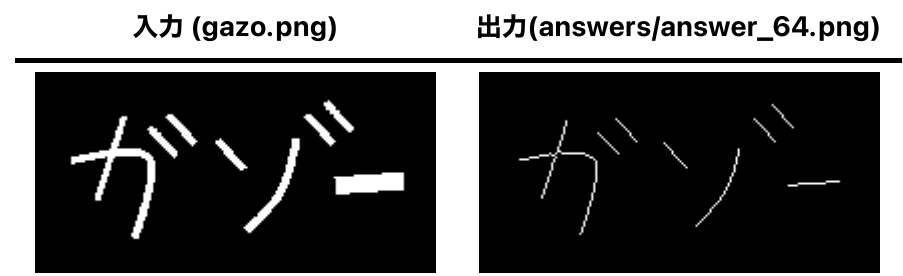
gazo.png にヒルディッチの細線化を行え。

アルゴリズムは、次の通り。

1. 左上からラスタスキャンする。
2. $x_0(x,y)=0$ ならば、処理なし。 $x_0(x,y)=1$ ならば次の5条件を満たす時に $x_0=-1$ に変える。
 1. 注目画素の4近傍に0が一つ以上存在する
 2. x_0 の8-連結数が1である
 3. $x_1 \sim x_8$ の絶対値の合計が2以上
 4. x_0 の8近傍に1が1つ以上存在する
 5. $x_n(n=1 \sim 8)$ 全てに対して以下のどちらかが成り立つ
 - x_n が-1以外
 - x_n を0とした時、 x_0 の8-連結数が1である
3. 各画素の-1を0に変える
4. 一回のラスタスキャンで3の変更数が0になるまで、ラスタスキャンを繰り返す。

入力 (*gazo.png*)

出力(*answers/answer_64.png*)



答え >> [answers/answer_64.py](#)

Q.65. Zhang-Suenの細線化

gazo.pngにZhang-Suenの細線化を行え。

ただし、以下の操作は全て0が線、1が背景とするので、gazo.pngの値を反転させる必要があることに注意。

注目画素x1(x,y)に対して8近傍を次のように定義する。

```
x9 x2 x3
x8 x1 x4
x7 x6 x5
```

これらに対して二つのステップを考える。

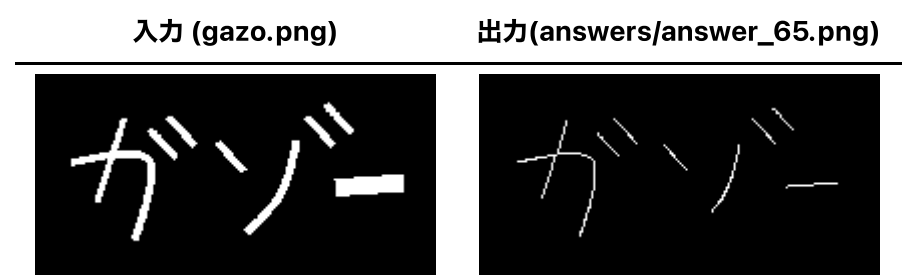
Step.1 ラスタスキャンを行い、以下の5条件を満たすピクセルを全て記録する。

- 1. 黒画素である
- 2. x2, x3, ..., x9, x2と時計まわりに見て、0から1に変わる回数がちょうど1
- 3. x2, x3, ..., x9の中で1の個数が2以上6以下
- 4. x2, x4, x6のどれかが1
- 5. x4, x6, x8のどれかが1 記録したピクセルを全て1に変更する。

Step.2 ラスタスキャンを行い、以下の5条件を満たすピクセルを全て記録する。

- 1. 黒画素である
- 2. x2, x3, ..., x9, x2と時計まわりに見て、0から1に変わる回数がちょうど1
- 3. x2, x3, ..., x9の中で1の個数が2以上6以下
- 4. x2, x4, x8のどれかが1
- 5. x2, x6, x8のどれかが1 記録したピクセルを全て1に変更する。

Step1, 2で変更する点が無くなるまで交互に繰り返す。



答え >> [answers/answer_65.py](#)

Q.66. HOG (Step.1) 勾配強度・勾配角度

*imori.jpg*のHOG特徴量の勾配強度・勾配角度を求めよ。

HOG(Histogram of Oriented Gradients)とは画像の特徴量表現の一種である。

特徴量とは画像の状態などを表すベクトル集合のことである。

画像認識(画像が何を写した画像か)や検出(画像の中で物体がどこにあるか)では、(1)画像から特徴量を得て(特徴抽出)、(2)特徴量を基に認識や検出を行う(認識・検出)。

ディープラーニングでは特徴抽出から認識までを機械学習により自動で行うため、HOGなどは見られなくなっているが、ディープラーニングが流行る前まではHOGは特徴量表現としてよく使われたらしい。

HOGは以下のアルゴリズムで得られる。

1. 画像をグレースケール化し、x、y方向の輝度勾配を求める

```
x方向: gx = I(x+1, y) - I(x-1, y)
y方向: gy = I(x, y+1) - I(x, y-1)
```

2. gx, gyから勾配強度と勾配角度を求める。

```
勾配強度: mag = sqrt(gx ** 2 + gy ** 2)
勾配角度: ang = arctan(gy / gx)
```

3. 勾配角度を [0, 180]で9分割した値に量子化する。つまり、[0,20]には0、[20, 40]には1というインデックスを求める。
4. 画像をN x Nの領域に分割し(この領域をセルという)、セル内で3で求めたインデックスのヒストグラムを作成する。ただし、当表示は1でなく勾配角度を求める。
5. C x Cのセルを1つとして(これをブロックという)、ブロック内のセルのヒストグラムを次式で正規化する。これを1セルずつずらしながら行うので、一つのセルが何回も正規化される。

```
h(t) = h(t) / sqrt(Sum h(t) + epsilon)
通常は epsilon=1
```

以上でHOG特徴量が求められる。


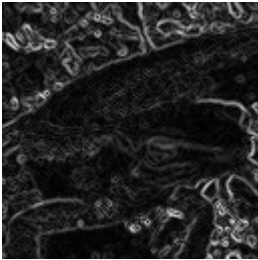
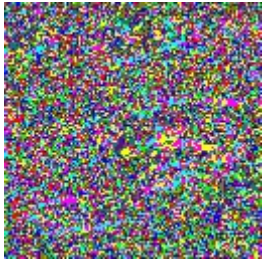
ここでは1から3までを行う。

解答例はみやすくするため、graは色付けしてある。またmagは[0, 255]に正規化してある。

入力 (imori.jpg)

勾配強度
(answers/answer_66_mag.jpg)

勾配角度
(answers/answer_66_gra.jpg)

入力 (imori.jpg)	勾配強度 (answers/answer_66_mag.jpg)	勾配角度 (answers/answer_66_gra.jpg)
		

答え >> [answers/answer_66.py](#)

Q.67. HOG (Step.2) 勾配ヒストグラム


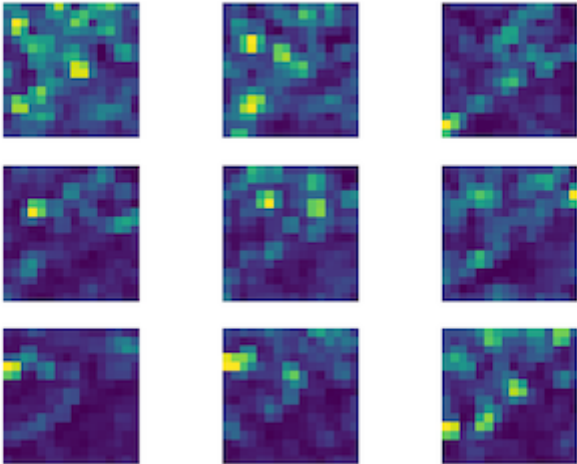
ここではHOGの4を実装する。

N=8として、8x8の領域を1セルとして、勾配角度のインデックスに勾配強度を投票する形式でヒストグラムを作成せよ。

解答は

```
1 2 3
4 5 6
7 8 9
```

の順に量子化したインデックスに対応するヒストグラムを示す。

入力 (imori.jpg)	出力(answers/answer_67.png)
	

答え >> [answers/answer_67.py](#)

Q.68. HOG (Step.3) ヒストグラム正規化

ここではHOGの5を実装する。

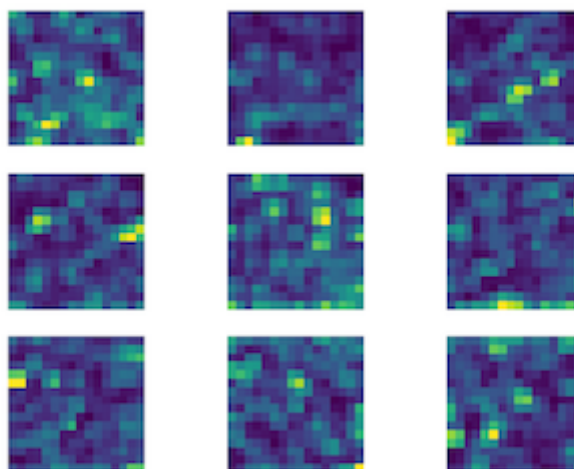
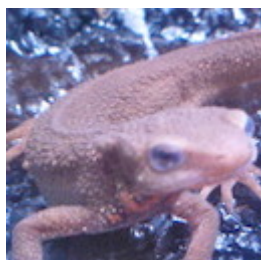
$C = 3$ として、 3×3 のセルを1ブロックとして扱い、ヒストグラムの正規化を行え。

```
h(t) = h(t) / sqrt(Sum h(t) + epsilon)
通常は epsilon=1
```

これでHOG特徴量が得られた。

入力 (imori.jpg)

出力(answers/answer_68.png)



答え >> [answers/answer_68.py](#)

Q.69. HOG (Step.4) 特徴量の描画

ここでは得られた特徴量を描画せよ。

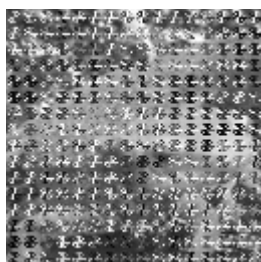
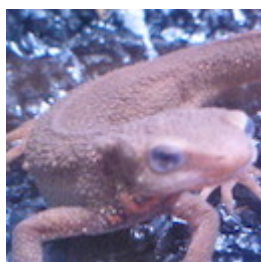
描画はimori.jpgをグレースケール化したものに重ねれば見やすい。

方法としては、セル内のインデックスごとに角度がついて直線を描けばよく、ヒストグラムの値が大きいほど白、値が小さいほど黒で描くと見やすい。

解答例

入力 (imori.jpg)

出力(answers/answer_69.jpg)



答え >> [answers/answer_69.py](#)

Q.70. カラートラッキング

*imori.jpg*に対してHSVを用いて青色の箇所のみが255となる画像を作成せよ。

カラートラッキングとは特定の色の箇所を抽出する手法である。

ただし、RGBの状態では色成分を指定するのは 256^3 のパターンがあり、とても大変である（というか手動ではかなり難しい）ので、HSV変換を用いる。

HSV変換とは Q.5で用いた処理であるが、RGBをH(色相)、S(彩度)、V(明度)に変換する手法である。

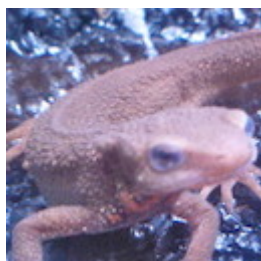
- Saturation(彩度) 彩度が小さいほど白、彩度が大きいほど色が濃くなる。 $0 \leq S \leq 1$
- Value (明度) 明度が小さいほど黒くなり、明度が大きいほど色がきれいになる。 $0 \leq V \leq 1$
- Hue(色相) 色を $0 \leq H \leq 360$ の角度で表し、具体的には次のように表される。

赤	黄色	緑	水色	青	紫	赤
0	60	120	180	240	300	360

つまり、青色のカラートラッキングを行うにはHSV変換を行い、 $180 \leq H \leq 260$ となる位置が255となるような二値画像を出力すればよい。

入力 (*imori.jpg*)

出力(*answers/answer_70.png*)



答え >> [answers/answer_70.py](#)