ラベリング処理アルゴリズムを利用した斑点の重心の計算

鈴木賢治 *

2023 年 9 月 12 日

1　前処理

ノイズや背景に勾配があることを想定し，測定された斑点像のラベリング処理をする前に，平滑化処理および二階微分画像を反転して前処理を行う．さらに，閾値で二値化して斑点部分を抽出する．

ここまでが前処理となる．その他にスパイクノイズなどがあれば，メジアン処理などを施す必要がある．

2　ラベリング

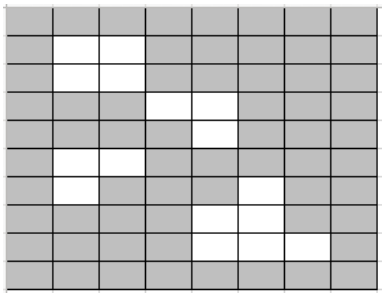

図 1: 二値化画像

図 1 のように，二値化画像処理された画像の白の部分 (または黒の部分) が連続した画素に同じ番号を割り振る処理をラベリング (labelling) と言う．通常，同じ番号ごとの面積 (画素数) や幅，高さ，重心などの特徴量を求めて回折位置，欠陥抽出や分類処理などに用いられる．

0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0
0	1	1	0	0	0	0	0
0	0	0	2	2	0	0	0
0	0	0	0	2	0	0	0
0	3	3	0	0	0	0	0
0	3	0	0	0	4	0	0
0	0	0	4	4	4	0	0
0	0	0	0	4	4	4	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0
0	1	1	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	0	0	1	0	0	0
0	2	2	0	0	0	0	0
0	2	0	0	0	3	0	0
0	0	0	3	3	3	0	0
0	0	0	0	3	3	3	0
0	0	0	0	0	0	0	0

図 2: ラベリング処理における 4 連結と 8 連結の例

図 2 に示すように，ラベリング処理の方法には 4 連結と 8 連結の 2 つの方法がある．ここでは，斑点形は円に近いものと想定し，4 連結アルゴリズムを選択する．

*新潟大学名誉教授 (電力中央研究所 EX 研究本部 客員研究員)

1. 配列 L を用意して，すべて 0 値を入力.
2. 左から右に以下の 3, 4 項に従いラスタースキャンをする.
3. 注目画素の上と左の画素が黒 (0 値) かつ白 (1 値) のとき，配列 L に +1 のラベル値を入れる．もしも，上と左の画素が 0 以外の番号があるときは，小さい番号を配列 L に入力する.
4. ルックアップテーブル LT にも同様にラベリング番号を入れる.

これを全スキャンする.

0	0	2	0	0	0
3	0	0	0	0	0
0	0	4	0	5	0
0	6	4	4	4	0
0	0	4	4	4	0
0	0	0	0	0	0

図 3: 更新すべきラベリング画素

さて，前述の方法でラベリングすると，図 3 に示すように同じ領域に別のラベル番号を割り当ててしまうことが必然的に現れる．そこで，ルックアップテーブル LT の更新をする．番号付を改める必要がある.

1. L に対して同様に左から右にスキャンし，注目の画素の上または右隣のラベルが小さければ，ルックアップテーブル LT の値をその小さい番号に更新する.
2. ラベリング更新に該当する配列に小さい番号を再入力して書き換える.
3. 以上を施して全スキャニングした後に，更新されたルックアップテーブル LT に従い，配列 L の番号付けをする.

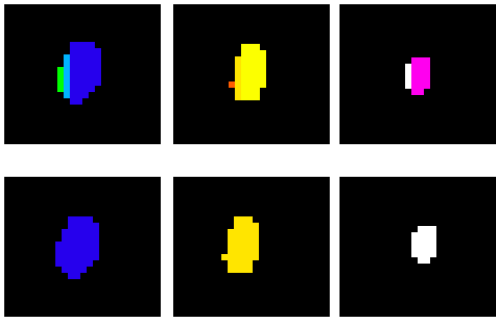

図 4: ラベリング処理された斑点

以上のラベリング処理により，まず図 4 の上に 3 種類の斑点がある．ルックアップテーブルを補正しないで機械的にラベリングされた結果である．図 3 で説明したような結果になっている．さらに，ルックアップテーブル LT の更新処理を行うと．図 4 の下段が得られる.

この結果で，同一領域の識別ができているが，ラベリングした番号は，更新により消えているのでラベリング番号は連番になっていない．それを解決するために，ファイル出力した後に画素のない斑点を除外して，読み込んで再度番号をつけ直す操作をしました.

その結果は，ラベリングしたテキスト画像 L.txt で保存される．また，斑点ごとの構成画素のリストは spot.txt として出力される.

3　重心の計算

各斑点の重心 (x_g, y_g) を求めるには，構成画素の n 個について，次の計算を行う．

$$x_g = \frac{1}{\sum_{i=1}^{n} I_i} \sum_{i=1}^{n} x_i I_i, \qquad y_g = \frac{1}{\sum_{i=1}^{n} I_i} \sum_{i=1}^{n} y_i I_i \tag{1}$$

ただし，I はその画素の輝度であり，x_i はその画素の x 座標，y_i も同様に画素の y 座標である．

重心は spot_list.txt としてファイル出力され，smooth.txt には重心位置に十字印を上書きしている．

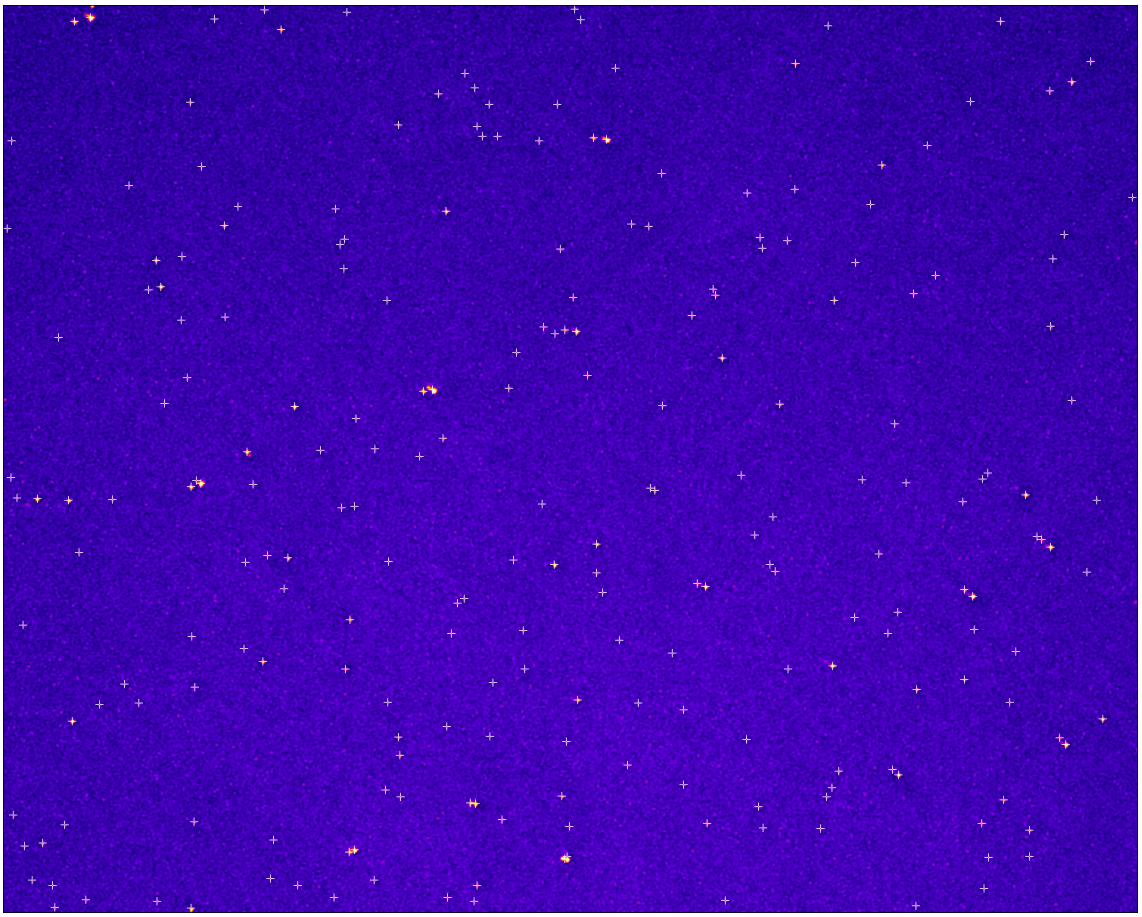

図 5: iPhone13 で測定した星空写真のラベリング処理と重心位置 (232 個)

図 5 にその結果を示す．閾値以上の 232 個について重心を計算し，重心位置に十字マークをつけた．これだけの処理ができれば，斑点像の二重露光法の処理に十分である．

付録　search_peak.f90 プログラム

```
!*********************************************************************
!*                        斑点抽出プログラム     search_peak.f90
!   Kenji Suzuki,
! Niigata University & Central Research Institute of Electric Power Industry
!*********************************************************************

program search_peak
  implicit none

  integer:: i,j,k,l,m,n,w,h,cnt,np,px(200),py(200),sm,nspot
  real*8:: x(0:2000,0:2000),xx(0:2000,0:2000)=0.0
  integer:: bi(0:2000,0:2000)=0
  real*8:: dx(0:2000,0:2000)=0.0,dI(0:2000,0:2000)=0.0,xf,yf,xyf,yxf
  real*8:: s,threshold

  w=1499;h=1199  ! 画像サイズ                    w=0:1499 h=0:1199
  m=5   !ピーク抽出セグメントのサイズ全幅 奇数
  n=(m-1)/2 ! 差分幅
  threshold=20.0 ! ピーク検出閾値

  open(unit=11,file='star1.txt',status='old')
    do j=0,h
      read(11,*) (x(i,j),i=0,w)
    end do
  close(11)

  !セグメントサイズで平滑化 3x3
  sm=1  !平滑化サイズ+/-1
  do j=sm,h-sm
    do i=sm,w-sm
      s=0.0; cnt=0
      do l=j-sm,j+sm
        do k=i-sm,i+sm
          s = s + x(k,l); cnt = cnt+1
        end do
      end do
      xx(i,j)=s/cnt
    end do
  end do
  !------------------------- 平滑化画像 出力 -------------------------
  open(unit=12,file='smooth.txt',status='unknown')
    do j=0,h
      write(12,120) (xx(i,j),i=0,w)
    end do
  close(12)

  !--------------------------------------------------------------
  np=0
  do j=0+n,h-n
    do i=0+n,w-n
      xf = (xx(i+n,j) + xx(i-n,j) -2.0*xx(i,j))/float(n)**2
      yf = (xx(i,j+n) + xx(i,j-n) -2.0*xx(i,j))/float(n)**2
      xyf = (xx(i+n,j+n) + xx(i-n,j-n) -2.0*xx(i,j))/float(n)**2
      yxf= (xx(i+n,j-n) + xx(i-n,j+n) -2.0*xx(i,j))/float(n)**2
      dx(i,j) = -xf-yf-xyf-yxf
    end do
  end do

  !セグメントサイズで平滑化 3x3
    sm=2  !平滑化サイズ+/-2
  do j=sm,h-sm
    do i=sm,w-sm
```

```
      s=0.0; cnt=0
      do l=j-sm,j+sm
        do k=i-sm,i+sm
          s = s + dx(k,l); cnt = cnt+1
        end do
      end do
      dI(i,j)=s/cnt
    end do
  end do

  open(unit=12,file='dI2.txt',status='unknown')
    do j=0,h
      write(12,120) (dI(i,j),i=0,w)
    end do
  close(12)

  do j=m,h-m
    do i=m,w-m
      if (dI(i,j) >17.0 ) then
        bi(i,j)=1
      else
        bi(i,j)=0
      end if
    end do
  end do

  open(unit=12,file='bi.txt',status='unknown')
    do j=0,h
      write(12,121) (bi(i,j),i=0,w)
    end do
  close(12)
  ! Labelling ==============================
  call labelling(w,h,nspot)

  call centroid(w,h,nspot)


!---------------------
  120 format(1542f8.1)
  121 format(1542I3)
end program search_peak

!************************************ ラベリング処理 ********************
subroutine labelling(w,h,m)  !画像の幅 w, 高さ h
  implicit none
  integer::w,h,i,j,bi(0:2000,0:2000),L(0:2000,0:2000)=0
  integer:: lk(1500) !  look up table
  integer:: label,k,cnt,sn,nos,m,n,ix,iy
  integer:: x(1500,300),y(1500,300) ,lbl(1500)

  open(unit=11,file='bi.txt',status='old')
    do j=0,h
      read(11,*) (bi(i,j),i=0,w)
    end do
  close(11)

  label=0
  do j=1,h-1
    do i=1,w-1
      if(bi(i,j) == 1) then !-------------

        if ((L(i,j-1) == 0) .and. (L(i-1,j)==0)) then
          label = label + 1
          L(i,j)=label    ! labelling
```

```
        lk(label)=label ! look up table
      end if

      if ((L(i,j-1) .ne. 0).and.(L(i-1,j) == 0)) then
        L(i,j)=L(i,j-1)
      end if

      if ((L(i-1,j) .ne. 0).and.(L(i,j-1) == 0)) then
        L(i,j)=L(i-1,j)
      end if

      if ((L(i-1,j) .ne. 0).and.(L(i,j-1) .ne. 0)) then
        if(L(i-1,j) > L(i,j-1)) then
          L(i,j)=L(i,j-1)
        else
          L(i,j)=L(i-1,j)
        end if
      end if

    end if !-------------------------------

  end do
end do

  open(unit=12,file='tmp.txt',status='unknown')
  do j=0,h
    write(12,*) (L(i,j),i=0,w)
  end do
close(12)

! ------------------------- look up table の更新 ------------------------

cnt=0   !cnt は, なんのため?
do j=1,h-1
  do i=1,w-1
    if (L(i,j).ne.0) then
      if ( (L(i,j-1).ne.0) .and. (L(i,j-1) < L(i,j)) ) then
        cnt=cnt+1
        lk(L(i,j))=L(i,j-1)
        L(i,j)=L(i,j-1)
      end if
      if ( (L(i+1,j).ne.0) .and. (L(i+1,j) < L(i,j)) ) then
        cnt=cnt+1
        lk(L(i,j))=L(i+1,j)
        L(i,j)=L(i+1,j)
      end if
    end if
  end do
end do

do i=1,label
  print *,i,lk(i)
end do

do j=1,h-1
  do i=1,w-1
    k=L(i,j)
    if (k .ne. 0) then
      L(i,j)=lk(k)
    end if
    !print *,L(i,j)
  end do
end do
```

```
open(unit=12,file='tmp.txt',status='unknown')
  do j=0,h
    write(12,*) (L(i,j),i=0,w)
  end do
close(12)


! output spot data   -------------------------

open (unit=14,file="tmp.txt", status="unknown")

m=0
do k=1,label

  sn=lk(k)
  cnt=0;

  do j=1,h-1
      do i=1,w-1
          if(L(i,j)==sn) then
            cnt=cnt+1
          end if
      end do
  end do

  if (cnt .ne. 0) then
      write(14,*) cnt
      m=m+1
  end if

  do j=1,h-1
      do i=1,w-1
          if(L(i,j)==sn) then
            write(14,*) i,j
            L(i,j)=0  !二度出力しないように初期化
          end if
      end do
  end do

end do

close(14)

print *,'Number of spots=',m

  open (unit=11,file="tmp.txt", status="old")
  do k=1,m
    read(11,*) lbl(k)
    do j=1,lbl(k)
      read(11,*) x(k,j),y(k,j)
    end do
  end do
  close(11)

! main spot data
  open (unit=12,file="spot.txt", status="unknown")

  write(12,*) m
  do k=1,m
    write(12,*) lbl(k)
    do j=1,lbl(k)
      write(12,*) x(k,j),y(k,j)
    end do
  end do
```

```
    close(12)

    ! output labelling
    ! initialize Label L
    do j=0,h
      do i=0,w
        L(i,j)=0
      end do
    end do

    do i=1,m
     n=lbl(i)
     do j=1,n
      ix= x(i,j) ; iy= y(i,j)
      L(ix,iy)=i
     end do
    end do

  open (unit=12,file="L.txt", status="unknown")
    do j=0,h
      write(12,*) (L(i,j),i=0,w)
    end do
  close(12)


    121 format(1542I4)
end subroutine labelling

! --------------------------- 各斑点の重心計算 ------------------
subroutine centroid(w,h,n)
  implicit none

  integer:: i,j,k,m,n,w,h,x,y
  real*8::dI(0:2000,0:2000),L(0:2000,0:2000),s,sx,sy,gx(1500),gy(1500)
  real*8:: imax

  !  各斑点の重心の計算
  open(unit=11,file='L.txt',status='old')

  do j=0,h
    read(11,*) (L(i,j),i=0,w)
  end do
  close(11)

  open(unit=11,file='dI2.txt',status='old')
  do j=0,h
    read(11,*) (dI(i,j),i=0,w)
  end do


  do k=1,n
    sx=0.d0; sy=0.d0;s=0.d0
    do j=1,h-1
      do i=1,w-1
        if (L(i,j) == k) then
          sx=sx+dI(i,j)*float(i)
          sy=sy+dI(i,j)*float(j)
          s=s+dI(i,j)
        end if
      end do
    end do
    gx(k)=sx/s; gy(k)=sy/s
    print *,k,gx(k),gy(k)
  end do
```

```
  ! ------------- 各斑点の重心リストの出力----------------
  open(unit=12,file='spot_list.txt',status='unknown')
    write(12,*) n
    do i=1,n
      write(12,'(I4,2X,F8.2,2X,F8.2)') i,gx(i),gy(i)
    end do
  close(12)


  ! ------------------ 斑点重心の印字 -----------------

  open(unit=11,file='smooth.txt',status='old')
  imax=0.0
  do j=0,h
    read(11,*) (dI(i,j),i=0,w)
  end do
  close(11)

  imax=maxval(dI)

  do k=1,n
    x=int(gx(k)); y=int(gy(k))

    do i=-5,-1
      if (dI(i+x,y) > 0) then
       dI(i+x,y)= imax !左線
      end if
    end do

    do i=1,5
      if (dI(i+x,y) < w) then
       dI(i+x,y)= imax !右線
      end if
    end do

    do j=-5,-1
      if (dI(x,j+y) > 0) then
       dI(x,j+y)= imax !上線
      end if
    end do

    do j=1,5
      if (dI(x,j+y) < h) then
       dI(x,j+y)= imax !下線
      end if
    end do

  end do

  open(unit=12,file='smooth.txt',status='old')
  do j=0,h
    write(12,*) (dI(i,j),i=0,w)
  end do
  close(12)
end subroutine centroid
```