義守大學電機所「電腦視覺」報告

單元三

影像切割 I — Thresholding

參考解答

MIAT(機器智慧與自動化技術)實驗室

中　華　民　國　93　年　10　月　18　日

1. 影像切割使用迭代式 **thresholding**

步驟 1、假設影像f(i,j)中的物件和背景的各自灰階統計分布均為高斯分布，任意取物件和背景的各自一點像素值作為物件和背景代表灰階的初始值。在真實影像中，可假設影像四個角落的像素為背景像素，其他則屬物件，已兩者的平均值μ_B 、μ_O作為背景和物件的初始灰階平均值。

步驟 2、計算$T = (\mu_B+\mu_O)/2$

步驟 3、以 T 為 threshold 將影像切割為物件和背景兩個區域， 再重新計算兩個區域的灰階平均值：

$$\mu_B = \frac{\sum_{(i,j)\in background} f(i,j)}{\#background\_pixels} \qquad \mu_O = \frac{\sum_{(i,j)\in objects} f(i,j)}{\#object\_pixels}$$

步驟 4、重回步驟 2,3，直到μ_B 、μ_O不再改變為止。

```
/*----------------------AlGORITHM--------------------------*
* Compute u1,the mean grey level of the corner pixels
* Compute u2,the mean grey level of all other pixels
* Told<=0
* Tnew<=(u1+u2)/2
* while Tnew!=Told do
*      u1<=mean grey level of pixels for which f(x,y)<Tnew
*      u2<=mean grey level of pixels for which f(x,y)>=Tnew
*      Told<=Tnew
*      Tnew=(u1+u2)/2
* end while
* Final threshold <= Tnew
*---------------------------------------------------------*/
```

程式碼：

```
#include <fstream.h>
#include "array.h"
#include "math.h"
int getOptimalThreshold(uc2D &grayimg);
void main()
{
  ifstream i("threshold.txt");
  ifstream in("finger300x300.raw",ios::binary);
  ofstream out("HW3-1-6.raw",ios::binary);
  uc2D ima;
  ima.Initialize(300,300);
  char c;
  int threshold=175;
```

```
   for(int i=0;i<ima.nr;i++)for(int j=0;j<ima.nc;j++)
   {
      in.get(c);ima.m[i][j]=c;
   }
//   threshold=getOptimalThreshold(ima);
     for(int i=0;i<ima.nr;i++)for(int j=0;j<ima.nc;j++)
     {
        if(ima.m[i][j]>threshold)ima.m[i][j]=255;
        else ima.m[i][j]=0;
     }
     for(int i=0;i<ima.nr;i++)for(int j=0;j<ima.nc;j++)
        out<<ima.m[i][j];
}
int getOptimalThreshold(uc2D &grayimg)
{
  ofstream o("threshold.txt");
  double u1,u2;
  int Told,Tnew;
  int t1,t2;
  float Threshold;
  u1=u2=0;
  t1=t2=0;
  for(int i=0;i<grayimg.nr;i++)
  {
    for(int j=0;j<grayimg.nc;j++)
    {
      if((i==0)||(i==grayimg.nr-1)||(j==0)||(j==grayimg.nc-1))
      {
        t1++;
        u1=u1+grayimg.m[i][j];
      }
      else
      {
        t2++;
        u2=u2+grayimg.m[i][j];
      }
    }
  }
```

```
Told=0;
u1=u1/t1;
u2=u2/t2;
Tnew=(u1+u2)/2;
while(Tnew!=Told)
{
   u1=0;
   u2=0;
   t1=0;
   t2=0;
   for(int i=0;i<grayimg.nr;i++)
   {
     for(int j=0;j<grayimg.nc;j++)
     {
       if(grayimg.m[i][j]<Tnew)
       {
         t1++;
         u1=u1+grayimg.m[i][j];
       }
       else
       {
         t2++;
         u2=u2+grayimg.m[i][j];
       }
     }
   }
   Told=Tnew;
   u1=u1/t1;
   u2=u2/t2;
   Tnew=(u1+u2)/2;
   o<<Tnew<<endl;
 }
 return int(Tnew);
```

圖 1.1 迭代第 1 次 **Threshold=**187

圖 1.2 迭代第 2 次 **Threshold=**182

圖 1.3 迭代第 3 次 **Threshold=**179

圖 1.4 迭代第 4 次 **Threshold=**177

圖 1.5 迭代第 5 次 **Threshold=**176

圖 1.6 迭代第 6 次 **Threshold=**175

2. Otsu 統計法決定影像切割的 Threshold

假設T為影像切割的threshold,則T所切割出來的背景和物件區域其灰階值分布將呈現C1, C2兩個聚類(cluster)。根據 Otsu，有兩個條件可判定 T 為最佳 threshold：

(1)C1 和 C2 的其間變異數(Between-variance)最大；

(2)C1 和 C2 的內在變異數(Within-variance)總合最小。

請參考單元二的標準差計算程式碼，設計一個程式，針對 finger300x300 影像，分別以條件(1)和條件(2)找出最佳 threshold，並比較影像切割結果。

(Between-variance 採用式 $BetweenVariance = |Mean1 - Mean2|$)

解答一：

程式碼片段：

```
double Mean1,Mean2;
double BetweenVariance;
double d1,d2;
double dev1,dev2;
double dMean;

double N1,N2;
int Threshold;

for(Threshold=20;Threshold<240;Threshold++)
{
    sum1=sum2=0;
    N1=N2=0;
    for(int i=0;i<sima.nr;i++)for(int j=0;j<sima.nc;j++)
    {
        if(sima.m[i][j]>Threshold)
        {
            N1++;
            sum1=sum1+(double)sima.m[i][j];
        }
        else
        {
            N2++;
            sum2=sum2+(double)sima.m[i][j];
        }
    }
    Mean1=sum1/(N1+0.0001);
    Mean2=sum2/(N1+0.0001);

    for(int i=0;i<sima.nr;i++)for(int j=0;j<sima.nc;j++)
    {
        if(sima.m[i][j]>Threshold)
        {
```

```
            d1=sima.m[i][j]-Mean1;
            sum1=sum1+d1*d1;
        }
        else
        {
            d2=sima.m[i][j]-Mean2;
            sum2=sum2+d2*d2;
        }
    }
    dev1=sqrt(sum1/(N1+0.0001));//C1 的 Within-variance
    dev2=sqrt(sum2/(N1+0.0001));//C1 的 Within-variance
    dMean=abs(Mean1-Mean2); //Between-variance
    cout<<Threshold<<"\t"<<dMean<<"\t"<<dev1<<"\t"<<dev2<<endl;
}
```

圖 2.1 Threshold=186

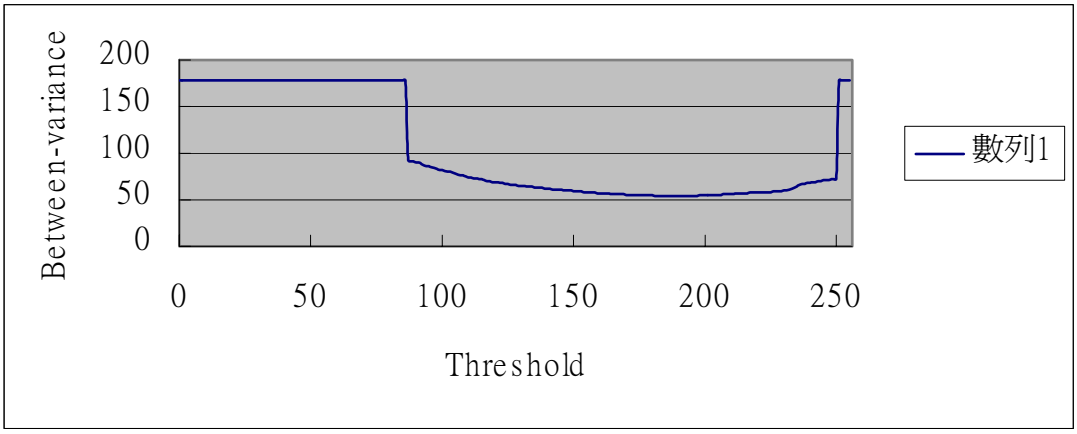
圖 2.2 Threshold=183

解答二(考慮 Heuristics constraints)：

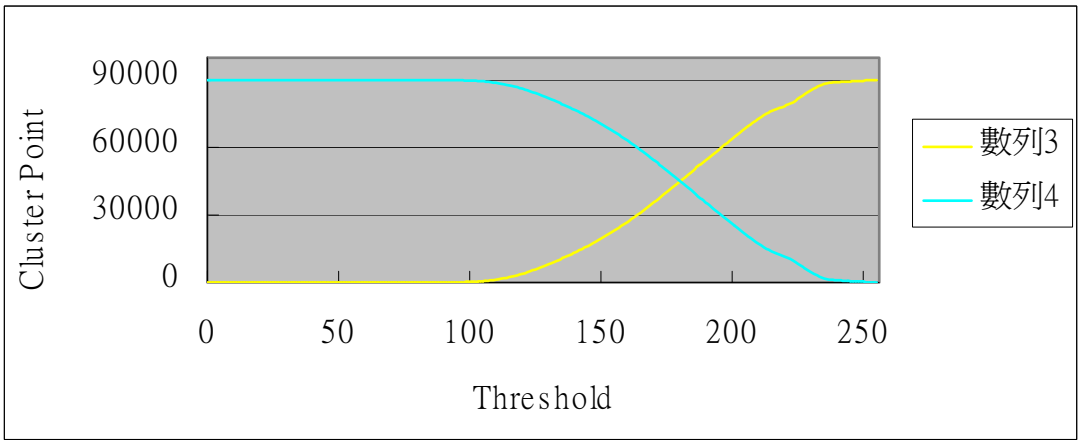
```
#include <fstream.h>
#include <math.h>
#include "array.h"

void get_Otsu_Threshold(uc2D& im,int &T1,int &T2)
{
    ofstream out("Otsu.xls");
    long N_C1,N_C2,sum_C1=0,sum_C2=0;
    float mean_C1,mean_C2;
    float variance_C1,variance_C2;
    int max_BV=0,min_WV=5000;   //MAX_Between-variance & MIN_Within-variance

    for(int T=0;T<256;T++)
    {
        N_C1=N_C2=0;
        sum_C1=sum_C2=0;
        for(int i=0;i<im.nr;i++)for(int j=0;j<im.nc;j++)
        {
            if(im.m[i][j]<T)
            {
```

```
                sum_C1 += im.m[i][j];
                N_C1++;
            }
            else
            {
                sum_C2 += im.m[i][j];
                N_C2++;
            }
        }

        mean_C1=(N_C1!=0)? (float)sum_C1/(float)(N_C1): 0; //Calculating the mean
        mean_C2=(N_C2!=0)? (float)sum_C2/(float)(N_C2): 0; //Calculating the mean
        float sumdev_C1=0.0,sumdev_C2=0.0;

        for(int i=0;i<im.nr;i++)for(int j=0;j<im.nc;j++)
        {
            if(im.m[i][j]<T)
            {
                int d1 =  im.m[i][j] - mean_C1;
                sumdev_C1 = sumdev_C1+ d1*d1;
            }
            else
            {
                int d2 =  im.m[i][j] - mean_C2;
                sumdev_C2 = sumdev_C2+ d2*d2;
            }
        }
        variance_C1 = (N_C1!=0)? sumdev_C1/N_C1: 0;//Calculating the variance
        variance_C2 = (N_C2!=0)? sumdev_C2/N_C2: 0;//Calculating the variance
        out<<T<<"\t"<<labs(mean_C1-mean_C2)<<"\t"<<(variance_C1+variance_C2)
             <<"\t"<<N_C1<<"\t"<<N_C2<<endl;

        if(labs(mean_C1-mean_C2)>max_BV &&
          (N_C1>0.25*im.nr*im.nc && N_C2>0.25*im.nr*im.nc))
        {
            max_BV=labs(mean_C1-mean_C2);
            T1=T;
        }
        if((variance_C1+variance_C2)<min_WV &&
          (N_C1>0.25*im.nr*im.nc && N_C2>0.25*im.nr*im.nc))
        {
            min_WV=variance_C1+variance_C2;
            T2=T;
        }
    }
}
void main(void)
{
    ifstream in("finger300x300.raw",ios::binary);
    ofstream outB("finger300x300_Between.raw",ios::binary);
    ofstream outW("finger300x300_Within.raw",ios::binary);
    uc2D ima;
    ima.Initialize(300,300);

    int T1,T2;

    char c;
    for(int i=0;i<ima.nr;i++)for(int j=0;j<ima.nc;j++)
    {
        in.get(c);ima.m[i][j]=c;
    }
```

```
    get_Otsu_Threshold(ima,T1,T2);

    for(int i=0;i<ima.nr;i++)for(int j=0;j<ima.nc;j++)
    {
        if(ima.m[i][j]<T1) outB<<(unsigned char) 0;
        else               outB<<(unsigned char) 255;

        if(ima.m[i][j]<T2) outW<<(unsigned char) 0;
        else               outW<<(unsigned char) 255;
    }
}
```

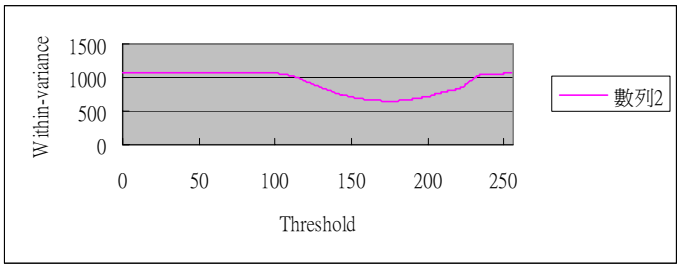


數列一為兩個聚類在各閥值情況下的期間變異數，若滿足條件一(即 C1 和 C2 的 Between-variance 最大)決定最佳閥值，則閥值將出現在 0~87、251~255 之間，顯然不合理。



數列三與數列四為兩個聚類所佔有的點(整張指紋影像 300*300=90000 點)，因此閥值將出現在 0~87、251~255 之間是無意義的。因此我們假設(Heuristics)指紋與背景兩個聚類必須大於整張影像的四分之一(閥值區間因而變為 150~200 之間)，所取得的閥值才有意義

條件一(C1 和 C2 的 Between-variance 最大)的情況最佳 Threshold=154



條件二(C1 和 C2 的 Within-variance 總合最小)的情況下最佳 Threshold=174