義守大學電機所「電腦視覺」報告

單元二

Histogram-based 影像增強

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　11　日

使用直方圖拓寬(histogram Stretching)影像對比增強。

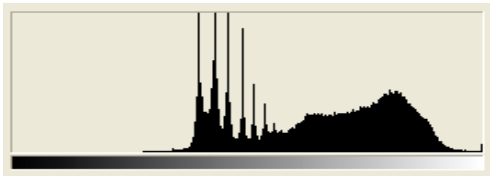
$$\text{Stretch}(I(r,c)) = \left[\frac{I(r,c) - I(r,c)_{\text{MIN}}}{I(r,c)_{\text{MAX}} - I(r,c)_{\text{MIN}}}\right][\text{MAX} - \text{MIN}] + \text{MIN}$$

$I(r,c)_{\text{MAX}}$ is the largest gray-level value in the image $I(r,c)$

$I(r,c)_{\text{MIN}}$ is the smallest gray-level value in $I(r,c)$

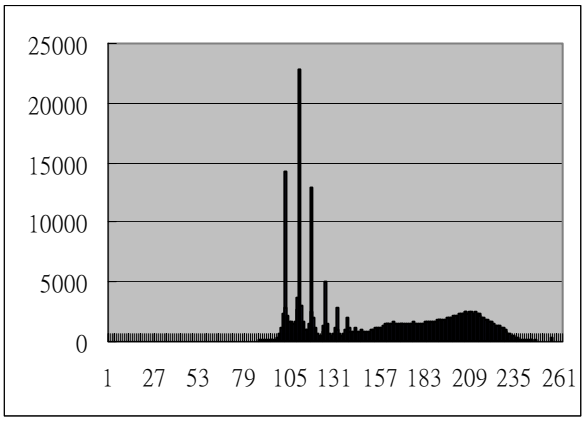
MAX and MIN correspond to the maximum and minimum gray-level values possible (for an 8-bit image these are 0 and 255)

如下圖將 kaoshiung512x512.raw 的灰階分布拉寬至[0,255]。



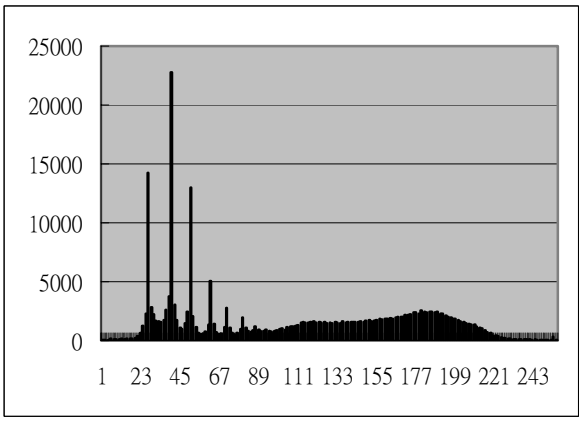
```
#include <fstream.h>
#include "array.h"
void main()
{
      ifstream in("kaoshiung512x512.raw",ios::binary);
      ofstream out("histogram Stretching.raw",ios::binary);
      ofstream out1("histogram.txt");

      uc2D ima;
      ima.Initialize(512,512);
      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;
      }
      int max=0,min=255;
      for(int i=0;i<ima.nr;i++)for(int j=0;j<ima.nc;j++)
      {
         if(ima.m[i][j]>max)max=ima.m[i][j];
         if(ima.m[i][j]<min)min=ima.m[i][j];
      }

      for(int i=0;i<ima.nr;i++)for(int j=0;j<ima.nc;j++)
         ima.m[i][j]=(float(ima.m[i][j]-min)/(max-min))*255;

      //histogram
      int histo[256];
      for(int i=0;i<256;i++)histo[i]=0;
      for(int i=0;i<ima.nr;i++)for(int j=0;j<ima.nc;j++)
         histo[ima.m[i][j]]++;
      for(int i=0;i<256;i++)
         out1<<i<<"\t"<<histo[i]<<endl;

      for(int i=0;i<ima.nr;i++)for(int j=0;j<ima.nc;j++)
         out<<ima.m[i][j];
}
```

程式執行結果：



(原始影像 histogram)

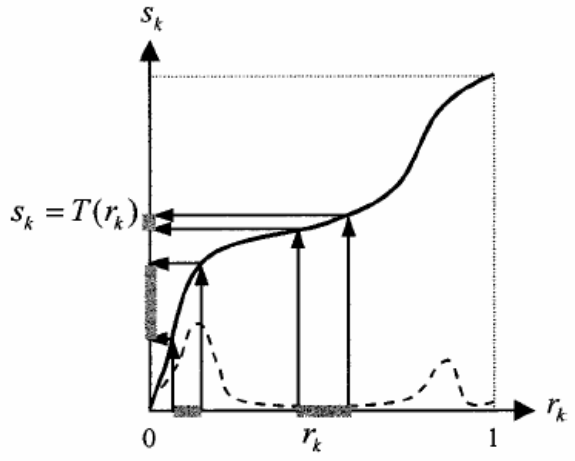
(原始影像)



(拓寬後影像 histogram)

(對比增強影像)

2.使用 Histogram Equalization(HE)增強影像對比



Transformation function for histogram equalization.

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j), \quad = \sum_{j=0}^{k} \frac{n_j}{n},$$
$$0 \le r_k \le 1 \quad \text{and} \quad k = 0, 1, \ldots, L-1$$

where

$p_r(r_j) = n_j/n$	probability density function (pdf) of the input image level j;
n	total number of pixels in the input image;
n_j	input pixel number of level j.

演算法：

Step 1. 計算影像灰階統計直方圖(histogram)Pr

Step 2. 從灰階統計直方圖計算累增直方圖(cumulative histogram) Sk

Step 3. 從累增直方圖計算等化分布直方圖(equalized histogram)f(x)，使灰階頻率平均分布在[X0, XL-1]:
f(x)=X0+(XL-1-X0)Sk

X0 是期望的最小灰階值(例如 0)，XL-1 是期望的最大灰階值(例如 255)

Step 4. 以此等化分布直方圖 f(x)當作映射函數，重新指定影像每一 pixel 的灰階值。

程式範例：

```
void HistogramEqualization(uc2D &ima0, uc2D &ima1)
{
   long ImaSize=ima0.nr*ima0.nc;
   int histo[256];  //histogram
   float accpbhisto[256]; // cumulative istogram
   int table[256];// Look-up table for mapping fuction of histogram equalization
// Initialize
   for(int i=0;i<256;i++)
   {
     histo[i]=0;
     table[i]=0;
     accpbhisto[i]=0.0;
   }
// Compute histogram
   for(int i=0;i<ima0.nr;i++)for(int j=0;j<ima0.nc;j++)histo[ima0.m[i][j]]++;

// Compute cumulative histogram
   accpbhisto[0]=float(histo[0])/float(ImaSize);
   for(int i=1;i<256;i++)
   {
      accpbhisto[i]=accpbhisto[i-1]+float(histo[i])/float(ImaSize);
   }
```

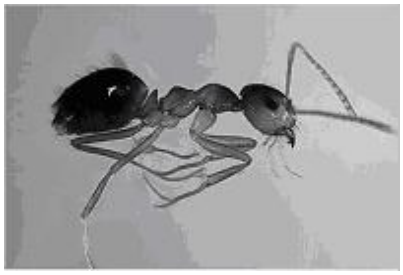
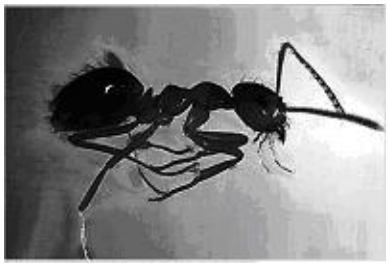
```
// compute mapping function
   for(int i=0;i<256;i++)table[i]=char(accpbhisto[i]*256.);
// Enhancement
    for(int i=0;i<ima0.nr;i++)for(int j=0;j<ima0.nc;j++)
      ima1.m[i][j]=table[ima0.m[i][j]];
}
```

程式碼：

```
#include <fstream.h>
#include <math.h>
#include "array.h"
void HistogramEqualization(uc2D &ima0, uc2D &ima1);
void main()
{
   ifstream in("kaoshiung512x512.raw",ios::binary);
   ofstream out("kaoshiung512x512HE.raw",ios::binary);
   uc2D ima2,ima3;
   ima2.Initialize(512,512);
   ima3.Initialize(512,512);
   char c;
   for(int i=0;i<ima2.nr;i++)for(int j=0;j<ima2.nc;j++)
   {
      in.get(c);ima2.m[i][j]=c;
   }
   HistogramEqualization(ima2,ima3);
   for(int i=0;i<ima2.nr;i++)for(int j=0;j<ima2.nc;j++)
      out<<ima3.m[i][j];
}
```

程式執行結果：

原始影像	HE 對比增強後的影像

3.Local HE 影像增強方法

每一個 pixel 與鄰近 pixel 的灰階值比較，決定其排序。再依此一排序的正比關係指定一個新的灰階值給這個 pixel。Local HE 影像增強方法是根據區域性(而非整張影像)的資訊來增強對比。

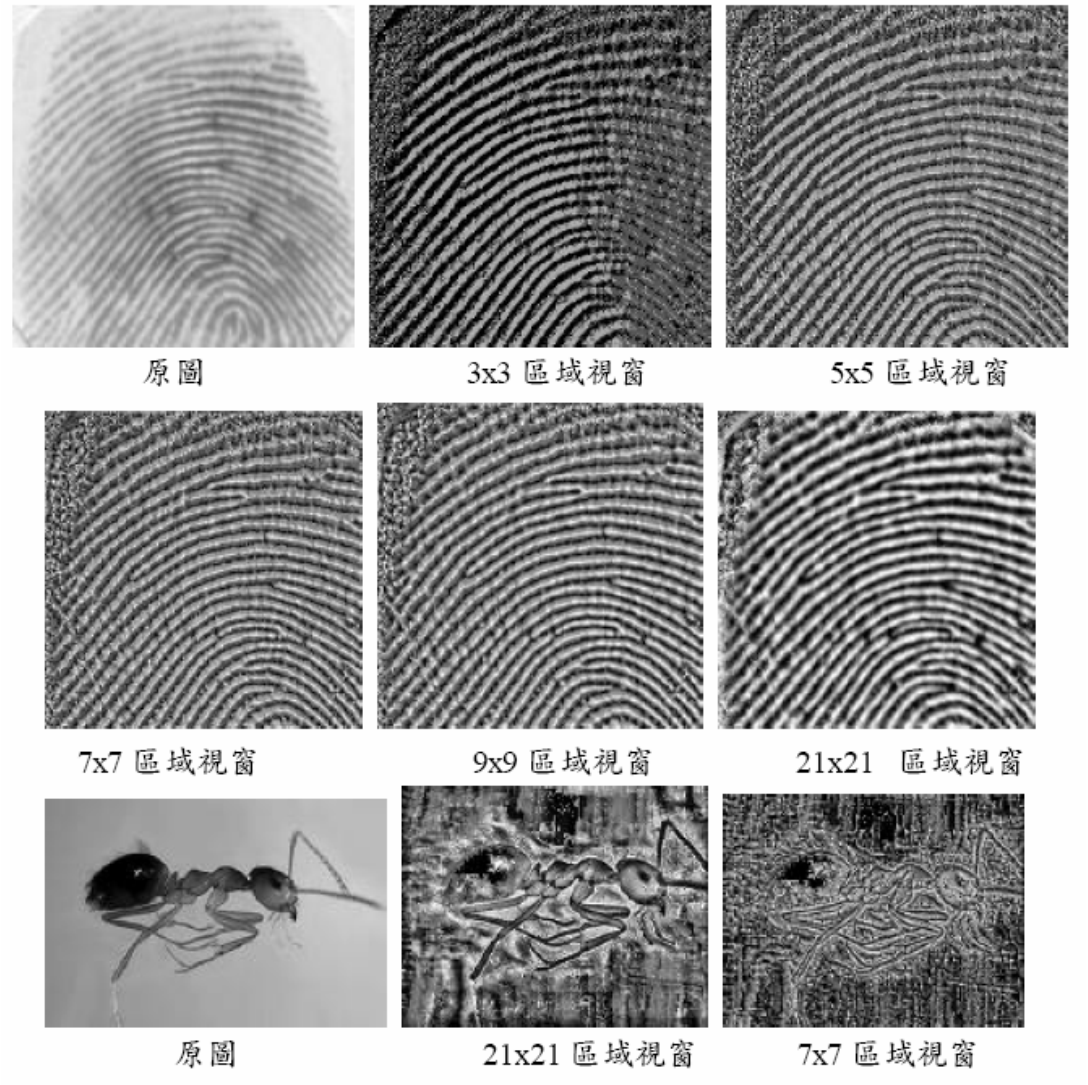
```
for each (x,y) in image do
{
     rank = 0
     for each (i,j) in contextual region of (x,y) do
     {
          if image[x,y] > image[i,j] then
               rank = rank + 1
     }
     output[x,y] = rank * max_intensity / (# of pixels in contextual region)
}
```

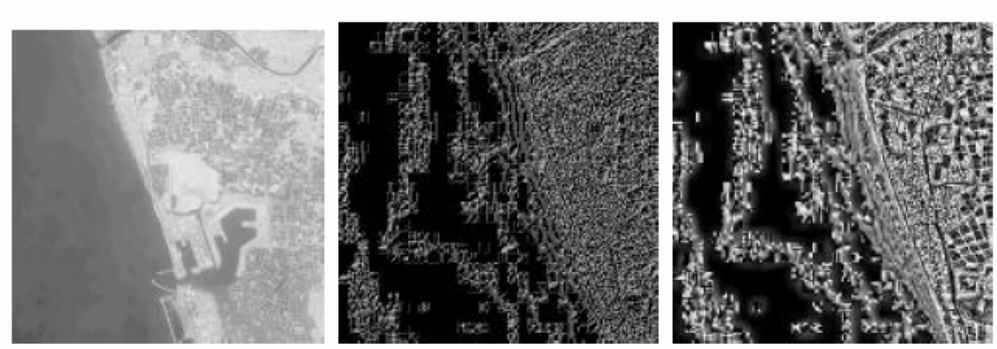
程式碼：

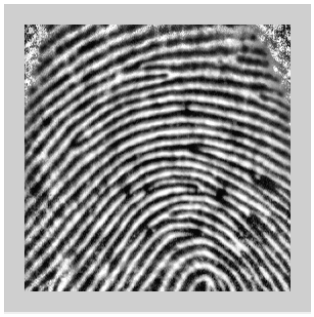
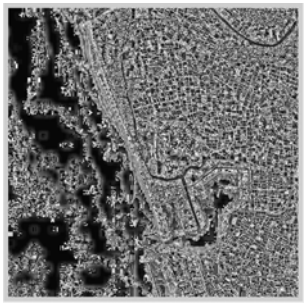
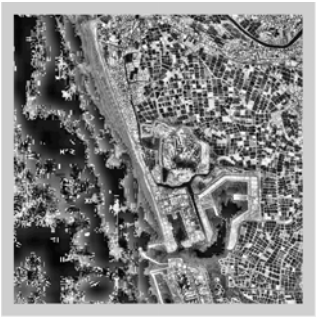
```
#include <fstream.h>
#include "array.h"
void LocalHE(uc2D &i1,uc2D &i2);
void main()
{
    ifstream in1("finger300x300.raw",ios::binary);
    uc2D ima0,ima1;
    ofstream out1("fingerlHE.raw",ios::binary);
   ima0.Initialize(300,300);
   ima1.Initialize(300,300);
   char c;
   for(int i=0;i<ima0.nr;i++)for(int j=0;j<ima0.nc;j++)
   {
      in1.get(c);ima0.m[i][j]=c;
   }
   LocalHE(ima0,ima1);
   for(int i=0;i<ima0.nr;i++)for(int j=0;j<ima0.nc;j++)out1<<ima1.m[i][j];
}
void LocalHE(uc2D &i1,uc2D &i2, int blocksize)
{
    int hsize,rank;
    if(blocksize%2==0) blocksize+=1;
    hsize=blocksize/2;
```

```
    int area= blocksize * blocksize;
    for(int i=hsize;i<i1.nr-hsize;i++)for(int j=hsize;j<i1.nc-hsize;j++)
    {
        rank=0;
        for(int k=i-hsize;k<=i+hsize;k++)for(int l=j-hsize;l<=j+hsize;l++)
        {
            if(i1.m[i][j]>i1.m[k][l])rank++;
        }
        i2.m[i][j]=rank*255/area;
    }
}
```

程式執行結果：





區域大小視窗 15X15	區域大小視窗 40X40	區域大小視窗 100X100
		
		
		

4. 參數可調整的 HE 影像增強方法—AHE(Adaptive Histogram Equalization)

$$\text{ACE} = k_1 \left[\frac{m_{I(r,c)}}{\sigma_l(r,c)} \right] \left[I(r,c) - m_l(r,c) \right] + k_2 \, m_l(r,c)$$

where $m_{I(r,c)}$ = is the mean for the entire image $I(r,c)$

σ_l = local standard deviation (in the window under consideration)

m_l = local mean (average in the window under consideration)

k_1, k_2 = constants, vary between 0 and 1

程式碼：

```
#include <fstream.h>
#include <iostream.h>
#include <math.h>
#include "array.h"
void  mean_stddev(uc2D &im, float &mean, float &std_dev);
void main()
{
   ifstream in("kaoshiung512x512.raw",ios::binary);
   ofstream out("test.raw",ios::binary);

   uc2D ima1,ima2,window;
   ima1.Initialize(512,512);
   ima2.Initialize(512,512);

   int i,j;
   for(i=0;i<ima1.nr;i++)for(j=0;j<ima1.nc;j++)ima1.m[i][j]=in.get();
   for(i=0;i<ima2.nr;i++)for(j=0;j<ima2.nc;j++)ima2.m[i][j]=ima1.m[i][j];

   int winsize=21, hsize=winsize/2;
   window.Initialize(winsize,winsize);
   float globalmean=0, mean=0.0;
   float std_dev=0.0;
   float k1=0.0;
   float k2=0.0;
   cout<<"input k1= ";
```

```
   cin>>k1;
   cout<<"input k2= ";
   cin>>k2;
   globalmean=0;
   for(i=0;i<ima1.nr;i++)for(j=0;j<ima1.nc;j++)globalmean+=ima1.m[i][j];
   globalmean= globalmean/( ima1.nr* ima1.nr);
   int ii,jj;
   int t;
   for(i=hsize;i<ima1.nr-hsize;i++)for(j=hsize;j<ima1.nc-hsize;j++)
   {
      for(ii=-hsize;ii<=hsize;ii++)for(jj=-hsize;jj<=hsize;jj++)
      {
        window.m[ii+hsize][jj+hsize]=ima1.m[i+ii][j+jj];
      }
       mean_stddev(window, mean, std_dev);
       t=(k1*(globalmean/std_dev)*(ima1.m[i][j]-mean))+ (k2*mean);
       if(t>255)ima2.m[i][j]=255;
       else ima2.m[i][j]=t;
   }
   for(i=0;i<ima2.nr;i++)for(j=0;j<ima2.nc;j++)
     out<<ima2.m[i][j];
}

void  mean_stddev(uc2D &im, float &mean, float &std_dev)
{
    int i, j;
    long N, sum=0;
    N = (long)(im.nr) * (long)(im.nc);
    for (i=0; i<im.nr; i++) for (j=0; j<im.nc; j++)
        sum += im.m[i][j];

    mean=(float)sum/(float)(N); //Calculating the mean

    float sumdev=0.0;
    float d=0.0;
    for (i=0; i<im.nr; i++) for (j=0; j<im.nc; j++)
    {
       d =  im.m[i][j] - mean;
```

```
      sumdev = sumdev+ d*d;
   }
   std_dev = sqrt(sumdev/N);//Calculating the standard deviance
}
```

程式執行結果：