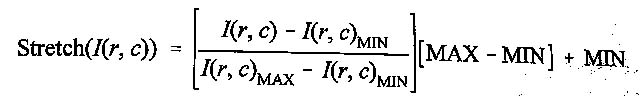
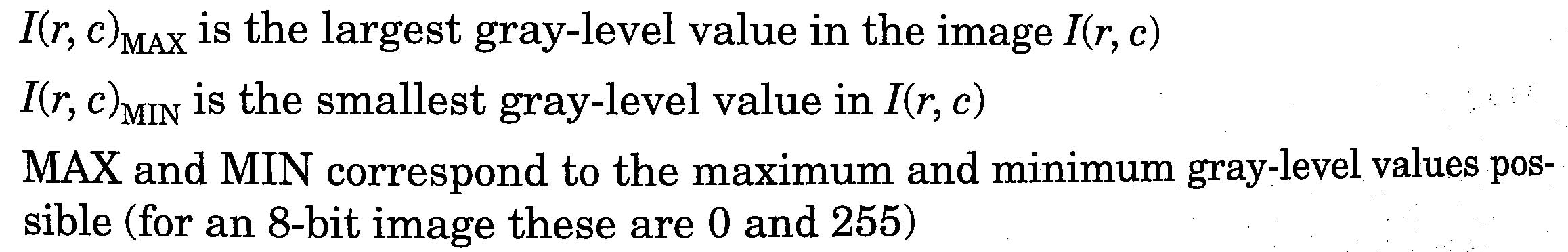
1. 使用直方圖拓寬(histogram Stretching)影像對比增強。



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

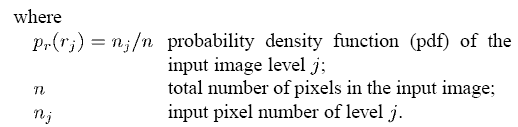
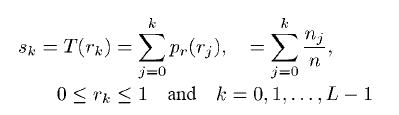
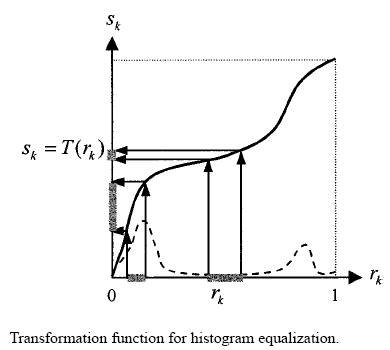
Source code :

|  |
| --- |
| #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 | 拓寬後的影像 |
| 1 | 2 |

1. 使用Histogram Equalization(HE)增強影像對比



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。

Source code :

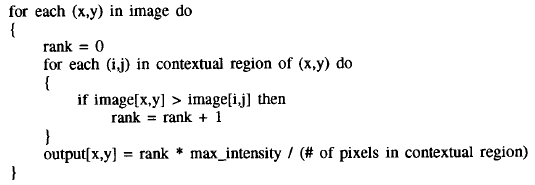
|  |
| --- |
| #include <iostream>  #include "stdlib.h"  #include "bmp.h"  void HistogramEqualization(unsigned char \*\*ima, unsigned char \*\*bima, int nr,int nc);  using namespace std;  int main(int argc, char\*\* argv) {    unsigned char \*\*ima, \*\*bima;  int nr,nc; //image height and width  char filename[128],temp;  bool isfilefine = false;  //read bmp image from file  cout << "Enter input filename:";  cin >> filename;  isfilefine = Read\_BMP(filename, ima, nr, nc);  if (!isfilefine) return 0;  bima=UC2D(nr,nc);  Write\_BMP\_8bits("ima.bmp", ima, nr, nc);  HistogramEqualization(ima,bima,nr,nc);  Write\_BMP\_8bits("ch2\_2.bmp", bima, nr, nc);  cout << "\nProgram done.\n";  return 1;  }  void HistogramEqualization(unsigned char \*\*ima, unsigned char \*\*bima, int nr,int nc)  {  long ImaSize=nr\*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<nr;i++)for(int j=0;j<nc;j++)histo[ima[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<nr;i++)for(int j=0;j<nc;j++)  bima[i][j]=table[ima[i][j]];  } |

程式執行結果：

|  |  |
| --- | --- |
| 原始影像  Hard3.bmp | HE對比增強後的影像histogram\_Stretching.bmp |
| Hard3 | histogram_Stretching |

1. 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。



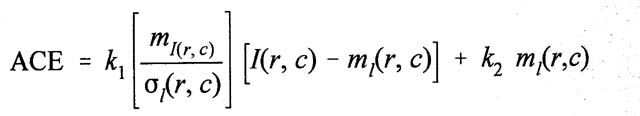
Source code :

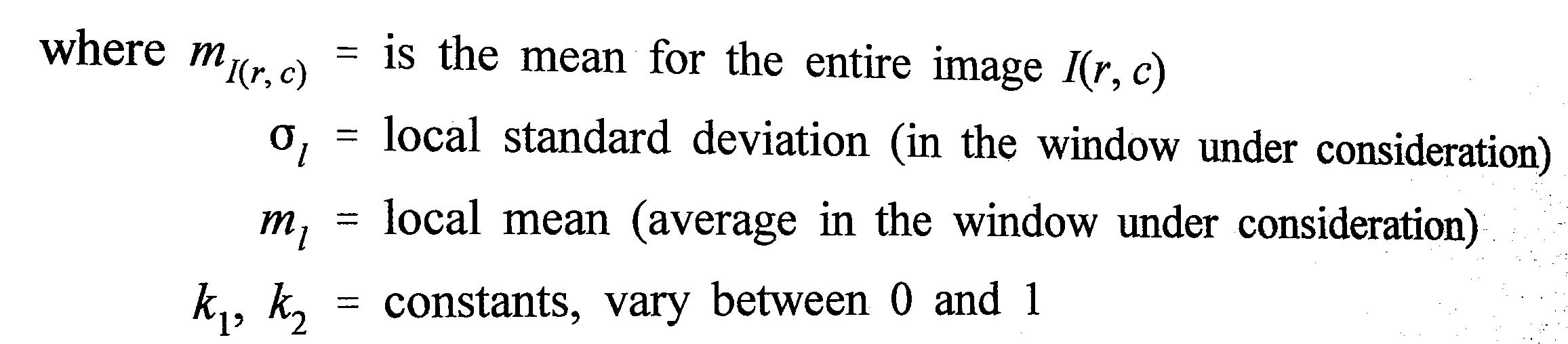
|  |
| --- |
| #include <iostream>  #include "stdlib.h"  #include "bmp.h"  void HistogramEqualization(unsigned char \*\*ima, unsigned char \*\*bima, int nr,int nc);  using namespace std;  int main(int argc, char\*\* argv) {    unsigned char \*\*ima, \*\*bima;  int nr,nc; //image height and width  char filename[128],temp;  bool isfilefine = false;    //read bmp image from file  cout << "Enter input filename:";  cin >> filename;  isfilefine = Read\_BMP(filename, ima, nr, nc);  if (!isfilefine) return 0;  bima=UC2D(nr,nc);  Write\_BMP\_8bits("ima.bmp", ima, nr, nc);  HistogramEqualization(ima,bima,nr,nc);  Write\_BMP\_8bits("ch2\_2.bmp", bima, nr, nc);  cout << "\nProgram done.\n";  return 1;  }  void HistogramEqualization(unsigned char \*\*ima, unsigned char \*\*bima, int nr,int nc)  {  long ImaSize=nr\*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<nr;i++)for(int j=0;j<nc;j++)histo[ima[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<nr;i++)for(int j=0;j<nc;j++)  bima[i][j]=table[ima[i][j]];  } |

程式執行結果(LocalHE.bmp) :

|  |  |  |
| --- | --- | --- |
| 區域大小視窗15X15 | 區域大小視窗40X40 | 區域大小視窗100X100 |
| LocalHE | LocalHE | LocalHE |

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





Source code :

|  |
| --- |
| #include <iostream>  #include <math.h>  #include "stdlib.h"  #include "bmp.h"  void mean\_stddev(unsigned char \*\*ima, float &mean, float &std\_dev, int nr, int nc);  int main(int argc, char\*\* argv) {  unsigned char \*\*ima, \*\*ahima, \*\*window;  int nr,nc; //image height and width  char filename[128], c;  bool isfilefine = false;  //read bmp image from file  cout << "Enter input filename:";  cin >> filename;  isfilefine = Read\_BMP(filename, ima, nr, nc);  if (!isfilefine) return 0;  ahima=UC2D(nr,nc);  Write\_BMP\_8bits("ima.bmp", ima, nr, nc);    for(int i=0;i<nr;i++)  for(int j=0;j<nc;j++)  ahima[i][j]=ima[i][j];  int winsize=21, hsize=winsize/2;  window=UC2D(winsize,winsize);  float globalmean=0, mean=0.0;  float std\_dev=0.0;  float k1=0.0, k2=0.0;  cout<<"input k1= ";  cin>>k1;  cout<<"input k2= ";  cin>>k2;  globalmean=0;  for(int i=0;i<nr;i++)  for(int j=0;j<nc;j++)  globalmean+=ima[i][j];  globalmean=globalmean/(nr\*nr);  int t;  for(int i=hsize;i<(nr-hsize);i++)  for(int j=hsize;j<(nc-hsize);j++)  {  for(int ii=-hsize;ii<=hsize;ii++)  for(int jj=-hsize;jj<=hsize;jj++)  window[ii+hsize][jj+hsize]=ima[i+ii][j+jj];  mean\_stddev(window, mean, std\_dev, winsize, winsize);  t=(k1\*(globalmean/std\_dev)\*(ima[i][j]-mean))+(k2\*mean);  if(t>255)  ahima[i][j]=255;  else  ahima[i][j]=t;  }  Write\_BMP\_8bits("AHE.bmp", ahima, nr, nc);    cout << "\nProgram done.\n";  return 1;  }  void mean\_stddev(unsigned char \*\*ima, float &mean, float &std\_dev, int nr, int nc)  {  long N, sum=0;  N=(long)nr\*(long)nc;  for(int i=0;i<nr;i++)  for(int j=0;j<nc;j++)  sum+=ima[i][j];  mean=(float)sum/(float)N; //Calculating the mean    float sumdev=0.0;  float d=0.0;  for(int i=0;i<nr;i++)  for(int j=0;j<nc;j++)  {  d=ima[i][j]-mean;  sumdev=sumdev+d\*d;  }  std\_dev=sqrt(sumdev/N); //Calculating the standard deviance  } |

程式執行結果(AHE.bmp)：

|  |  |  |
| --- | --- | --- |
| k1 = 0.1, k2 = 0.1 | k1 = 0.5, k2 = 0.5 | k1 = 0.9, k2 = 0.9 |
| AHE | AHE | AHE |
| k1 = 0.1, k2 = 0.9 | k1 = 0.3, k2 = 0.7 | k1 = 0.9, k2 = 0.1 |
| AHE | AHE | AHE |