# Digital Image Processing编程作业

学生姓名： 许一宁 学号： 2018222050154

### 第一章

编程1：编程将一幅图像降质为多个低空间分辨率图像

##### 源代码及重点语句的注释

|  |
| --- |
| void CImage\_ProcessingView::OnResolution(){  // TODO: 在此添加命令处理程序代码  if (m\_Image.IsNull()) return;//判断图像是否为空，对空图像操作会出现未知的错误  int w = m\_Image.GetWidth();//获得图像的宽度  int h = m\_Image.GetHeight();//获得图像的高度  int bits = m\_Image.GetBPP();  int C = 3;  int inter = 5; //降低图像五倍分辨率  for (int c = 0; c < C; c++){  for (int j = 0; j < h; j++){  for (int k = 0; k < w; k++){  m\_Image.m\_pBits[c][j][k] = m\_Image.m\_pBits[c][j / inter \* inter][k / inter \* inter];  }  }  }  Invalidate(1); //强制调用ONDRAW函数，ONDRAW会绘制图像  } |

##### 实现结果展示

|  |  |
| --- | --- |
| 原图 | 降低分辨率后的图 |

编程2：编程将一幅256级的灰度图像分解为不同灰度分辨率（128、64、32、16、8、4、2）的图像。

##### 源代码及重点语句的注释

|  |
| --- |
| void CImage\_ProcessingView::OnGraychange(){  // TODO: 在此添加命令处理程序代码  if (m\_Image.IsNull()) return;//判断图像是否为空，空图像进行操作会出现未知的错误  int w = m\_Image.GetWidth();//获得图像的宽度  int h = m\_Image.GetHeight();//获得图像的高度  int bits = m\_Image.GetBPP();  int hierarchical = 2;  for (int j = 0; j < h; j++){  for (int k = 0; k < w; k++){  int ave = 0.1140 \*m\_Image.m\_pBits[0][j][k] + 0.5870 \*m\_Image.m\_pBits[1][j][k] + 0.2989 \*m\_Image.m\_pBits[2][j][k]; // change image to gray  m\_Image.m\_pBits[0][j][k] = ave;  m\_Image.m\_pBits[1][j][k] = ave;  m\_Image.m\_pBits[2][j][k] = ave;  }  }  int K = 2; //灰度分辨率等级  for (int i = 0; i < h; i++){  for (int j = 0; j < w; j++){  int step = 256 / (K); //计算等级跨度  int value = (step << 1) - 1;  m\_Image.m\_pBits[0][i][j] = (m\_Image.m\_pBits[0][i][j] / step) \* value;  m\_Image.m\_pBits[1][i][j] = (m\_Image.m\_pBits[1][i][j] / step) \* value;  m\_Image.m\_pBits[2][i][j] = (m\_Image.m\_pBits[2][i][j] / step) \* value;  }  }  Invalidate(1); //强制调用ONDRAW函数，ONDRAW会绘制图像  } |

##### 实现结果展示

以下为原图以及分辨率依次为128,64,32,16,8,4,2。

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

编程3：编程实现图像差分、多幅图像相加去噪。

##### 流程图

|  |  |
| --- | --- |
| 图片差分流程图 | 图片叠加去噪流程图 |

##### 源代码及重点语句的注释

1、图片差分

|  |
| --- |
| void CImage\_ProcessingView::OnDiff(){  // TODO: 在此添加命令处理程序代码  if (m\_Image.IsNull()) return;//判断图像是否为空，对空图像操作会出现未知的错误  int w = m\_Image.GetWidth();//获得图像的宽度  int h = m\_Image.GetHeight();//获得图像的高度  int bits = m\_Image.GetBPP();  int hierarchical = 2;  MyImage\_ m\_Image2;  CFileDialog dlg(TRUE);//同样是打开一个新的对话框，存储别的输入图片  if (IDOK == dlg.DoModal()){  if (!m\_Image2.IsNull())  m\_Image2.Destroy();  m\_Image2.Load(dlg.GetPathName());  if (m\_Image2.IsNull())  return;  int w0 = m\_Image2.GetWidth();  int h0 = m\_Image2.GetHeight();  if (w0 != w || h0 != h) {//比较两张图片大小，如果不同就return  return;  }  for (int i = 0; i < h; i++){  for (int j = 0; j < w; j++){  m\_Image.m\_pBits[0][i][j] = m\_Image.m\_pBits[0][i][j] - m\_Image2.m\_pBits[0][i][j];  m\_Image.m\_pBits[1][i][j] = m\_Image.m\_pBits[1][i][j] - m\_Image2.m\_pBits[1][i][j];  m\_Image.m\_pBits[2][i][j] = m\_Image.m\_pBits[2][i][j] - m\_Image2.m\_pBits[2][i][j]; //逐像素计算差值  }  }  Invalidate(1);  }  } |

2、叠加去燥

|  |
| --- |
| void CImage\_ProcessingView::OnDenoiseing(){  // TODO: 在此添加命令处理程序代码  if (m\_Image.IsNull()) return;//判断图像是否为空，对空图像操作会出现未知的错误  int w = m\_Image.GetWidth();//获得图像的宽度  int h = m\_Image.GetHeight();//获得图像的高度  int bits = m\_Image.GetBPP();  int hierarchical = 2;  MyImage\_ m\_Image2;  CFileDialog dlg(TRUE);//同样是打开一个新的对话框，存储别的输入图片  dlg.m\_ofn.Flags |= OFN\_ALLOWMULTISELECT;//允许选择多个文件  dlg.m\_ofn.nMaxFile = 20 \* 101;//最多可以打开20个文件，每个文件名的字符数<=100  const DWORD numberOfFileNames = 32;//最多允许32个文件  const DWORD fileNameMaxLength = MAX\_PATH + 1;  const DWORD bufferSize = (numberOfFileNames \* fileNameMaxLength) + 1;  TCHAR\* filenamesBuffer = new TCHAR[bufferSize];  filenamesBuffer[0] = NULL;//必须的  filenamesBuffer[bufferSize - 1] = NULL;  dlg.m\_ofn.lpstrFile = filenamesBuffer;  dlg.m\_ofn.nMaxFile = bufferSize;  CStringArray strArrFilePaths;  if (dlg.DoModal() == IDOK){  int k = 0;  POSITION pos = dlg.GetStartPosition();//获取第一个文件名的位置  while (pos != NULL) //GetNextPathName()返回当前pos的文件名，并将下一个文件名的位置保存到pos中，最后达到连续叠加多幅图片{  if (!m\_Image2.IsNull())  m\_Image2.Destroy();  m\_Image2.Load(dlg.GetNextPathName(pos));  if (m\_Image2.IsNull())  return;  for (int i = 0; i < h; i++){  for (int j = 0; j < w; j++){  m\_Image.m\_pBits[0][i][j] = (m\_Image.m\_pBits[0][i][j] / (k + 2))\*(k + 1) + m\_Image2.m\_pBits[0][i][j] / (k + 2);  m\_Image.m\_pBits[1][i][j] = (m\_Image.m\_pBits[1][i][j] / (k + 2))\*(k + 1) + m\_Image2.m\_pBits[1][i][j] / (k + 2);  m\_Image.m\_pBits[2][i][j] = (m\_Image.m\_pBits[2][i][j] / (k + 2))\*(k + 1) + m\_Image2.m\_pBits[2][i][j] / (k + 2); //计算叠加图片的数量并继续叠加。  }  }  k++;  }  Invalidate(1);  }  } |

##### 实现结果展示

1. 差分结果图

|  |  |
| --- | --- |
| 原图 | 相减图像 |
| 差分后图像 | |

2、叠加去噪结果图。

|  |  |
| --- | --- |
| 加入噪声后的图片 | 叠加去噪后的效果 |

### 第三章

编程1：编写程序统计任意一幅图像的直方图并显示

##### 源代码及重点语句的注释

|  |
| --- |
| void CImage\_ProcessingView::OnHistogram(){  // TODO: 在此添加命令处理程序代码  if (m\_Image.IsNull()) return;//判断图像是否为空，对空图像操作会出现未知的错误  int w = m\_Image.GetWidth();//获得图像的宽度  int h = m\_Image.GetHeight();//获得图像的高度  int bits = m\_Image.GetBPP();  int hierarchical = 2;  if (bits == 24 || bits == 32) {  for (int i = 0; i < h; i++){  for (int j = 0; j < w; j++){  int ave = 0.1140 \*m\_Image.m\_pBits[0][i][j] + 0.5870 \*m\_Image.m\_pBits[1][i][j] + 0.2989 \*m\_Image.m\_pBits[2][i][j];  m\_Image.m\_pBits[0][i][j] = ave;  m\_Image.m\_pBits[1][i][j] = ave;  m\_Image.m\_pBits[2][i][j] = ave;  }  }  }  m\_Image.calcHistogram();//调用m\_Image内的直方图统计函数  paintHistDialog dlg(this);//用一个CImage\_ProcessingView的指针取初始化dlg  dlg.DoModal();  Invalidate(1);  }  //直方图统计函数  void MyImage\_::calcHistogram(void)//直方图统计{  BYTE \*lpSrc; //指向源图的指针  int w = GetWidth();  int h = GetHeight();  int arr[256] = {0}; //每个灰度值统计  int nrow = m\_CImage.GetPitch();//获得Image每一行像素的RGB所占用的存储空间的大小  for (int i = 0; i < h; i++){  for (int j = 0; j < w; j++){  BYTE value = m\_pBits[0][i][j];  arr[value] ++;  }  }  for (int k = 0; k < 256; k++){  hist[k] = arr[k] / (w\*h\*1.0f); //Normalization  }  } |

##### 实现结果展示

|  |  |
| --- | --- |
|  |  |

编程2：编程实现图像均衡化和图像规格化

##### 流程图

##### 源代码及重点语句的注释

1、直方图均衡

|  |
| --- |
| void CImage\_ProcessingView::OnHistequal(){  // TODO: 在此添加命令处理程序代码  if (m\_Image.IsNull()) return;//判断图像是否为空，对空图像操作会出现未知的错误  int w = m\_Image.GetWidth();//获得图像的宽度  int h = m\_Image.GetHeight();//获得图像的高度  int bits = m\_Image.GetBPP();  int hierarchical = 2;  if (bits == 24 || bits == 32) {  for (int i = 0; i < h; i++){  for (int j = 0; j < w; j++){  int ave = 0.1140 \*m\_Image.m\_pBits[0][i][j] + 0.5870 \*m\_Image.m\_pBits[1][i][j] + 0.2989 \*m\_Image.m\_pBits[2][i][j];  m\_Image.m\_pBits[0][i][j] = ave; // change image to gray  m\_Image.m\_pBits[1][i][j] = ave;  m\_Image.m\_pBits[2][i][j] = ave;  }  }  }  m\_Image.calcHistogram();  float s[256] = { 0 };//均衡  int hist\_equal[256] = { 0 };//均衡后  for (int i = 0; i < 256; i++){  for (int j = 0; j <= i; j++){  s[i] += 255 \* m\_Image.hist[j]; //建立均衡后的映射关系  }  hist\_equal[i] = floor(s[i]); // 对映射关系去下整数  }  for (int i = 0; i < h; i++){  for (int j = 0; j < w; j++){  int value = m\_Image.m\_pBits[0][i][j];  int new\_value = hist\_equal[value]; //利用映射关系对图片像素进行操作  m\_Image.m\_pBits[0][i][j] = new\_value;  m\_Image.m\_pBits[1][i][j] = new\_value;  m\_Image.m\_pBits[2][i][j] = new\_value;  }  }  m\_Image.calcHistogram();  paintHistDialog dlg(this);//用一个CImage\_ProcessingView的指针取初始化dlg  dlg.DoModal();  Invalidate(1);  } |

1. 直方图规定

|  |
| --- |
| void CImage\_ProcessingView::OnHistmatch(){  // TODO: 在此添加命令处理程序代码  MyImage\_ m\_Image\_Match;  CFileDialog dlg(TRUE);//同样是打开一个新的对话框，存储别的输入图片  if (IDOK == dlg.DoModal()) {  if (!m\_Image\_Match.IsNull())  m\_Image\_Match.Destroy();  m\_Image\_Match.Load(dlg.GetPathName());  if (m\_Image\_Match.IsNull())return;  int w = m\_Image\_Match.GetWidth();//获得图像的宽度  int h = m\_Image\_Match.GetHeight();//获得图像的高度  int arr[256] = { 0 };  int G[256] = { 0 };//存放待匹配图片的每个像素值的统计量  int bits = m\_Image\_Match.GetBPP();  if (bits == 24 || bits == 32){  for (int i = 0; i < h; i++){  for (int j = 0; j < w; j++){  int ave = 0.1140 \*m\_Image\_Match.m\_pBits[0][i][j] + 0.5870 \*m\_Image\_Match.m\_pBits[1][i][j] + 0.2989 \*m\_Image\_Match.m\_pBits[2][i][j];  m\_Image\_Match.m\_pBits[0][i][j] = ave; // change image to gray  m\_Image\_Match.m\_pBits[1][i][j] = ave;  m\_Image\_Match.m\_pBits[2][i][j] = ave;  }  }  }  for (int i = 0; i < h; i++){  for (int j = 0; j < w; j++){  BYTE value = m\_Image\_Match.m\_pBits[0][i][j];  arr[value] ++;//计算统计直方图  }  }  if (m\_Image.IsNull()) return;//判断图像是否为空，空图像操作会出现未知的错误  int w0 = w;  int h0 = h;  w = m\_Image.GetWidth();//获得图像的宽度  h = m\_Image.GetHeight();//获得图像的高度  for (int i = 0; i < 256; i++){  for (int j = 0; j <= i; j++){  G[i] += float(arr[j]) / w0 / h0 \* w \* h;  } //计算映射到图片中每个像素值的统计量  }  bits = m\_Image.GetBPP();  int hierarchical = 2;  int n[256] = { 0 };//存放源图片的每个像素值的统计量  int nMap[256] = { 0 };//存放源图到目标图片的像素值映射表  if (bits == 24 || bits == 32){  for (int i = 0; i < h; i++){  for (int j = 0; j < w; j++){  int ave = 0.1140 \*m\_Image.m\_pBits[0][i][j] + 0.5870 \*m\_Image.m\_pBits[1][i][j] + 0.2989 \*m\_Image.m\_pBits[2][i][j];  m\_Image.m\_pBits[0][i][j] = ave;  m\_Image.m\_pBits[1][i][j] = ave;  m\_Image.m\_pBits[2][i][j] = ave;  }  }  }  for (int i = 0; i < h; i++){  for (int j = 0; j < w; j++){  BYTE value = m\_Image.m\_pBits[0][i][j];  n[value] ++;//源图的统计直方图  }  }  int count\_g = 0;  int count\_n = 0;  int sum = 0;  while (true) {  if (count\_n == 256) break;  if (sum < G[count\_g]) {  //count\_n++;  sum += n[count\_n++];  continue;  }  else {  nMap[count\_n] = count\_g;  count\_g++;  }  if (count\_g == 255) {  nMap[255] = 255;  break;  }  }  int k = nMap[255];  for (int i = 255; i >= 0; i--){  if (nMap[i] == 0) nMap[i] = k;  else k = nMap[i];  }  for (int i = 0; i < h; i++){  for (int j = 0; j < w; j++)  int value = m\_Image.m\_pBits[0][i][j];  m\_Image.m\_pBits[0][i][j] = nMap[value];  m\_Image.m\_pBits[1][i][j] = nMap[value];  m\_Image.m\_pBits[2][i][j] = nMap[value];  }  }  m\_Image.calcHistogram();  paintHistDialog dlg1(this);//用一个CImage\_ProcessingView的指针取初始化dlg  dlg1.DoModal();  Invalidate(1);  }  } |

##### 实现结果展示

编程3：编写程序统计任意一幅图像的直方图并显示

##### 源代码及重点语句的注释

1、均值滤波

|  |
| --- |
| void CImage\_ProcessingView::OnMeanfilter(){  // TODO: 在此添加命令处理程序代码  if (m\_Image.IsNull()) return;//判断图像是否为空，对空图像操作会出现未知的错误  int w = m\_Image.GetWidth();//获得图像的宽度  int h = m\_Image.GetHeight();//获得图像的高度  int bits = m\_Image.GetBPP();  int hierarchical = 2;  MyImage\_ m\_Image2;  m\_Image2.Load(filename);  for (int i = 1; i < h - 1; i++){  for (int j = 1; j < w - 1; j++){//对像素点领域内的像素值进行相加求平均  m\_Image.m\_pBits[0][i][j] = m\_Image2.m\_pBits[0][i - 1][j - 1] / 9 + m\_Image2.m\_pBits[0][i][j - 1] / 9 + m\_Image2.m\_pBits[0][i + 1][j - 1] / 9 + m\_Image2.m\_pBits[0][i - 1][j] / 9 + m\_Image2.m\_pBits[0][i][j] / 9 + m\_Image2.m\_pBits[0][i + 1][j] / 9 + m\_Image2.m\_pBits[0][i - 1][j + 1] / 9 + m\_Image2.m\_pBits[0][i][j + 1] / 9 + m\_Image2.m\_pBits[0][i + 1][j + 1] / 9;  m\_Image.m\_pBits[1][i][j] = m\_Image2.m\_pBits[1][i - 1][j - 1] / 9 + m\_Image2.m\_pBits[1][i][j - 1] / 9 + m\_Image2.m\_pBits[1][i + 1][j - 1] / 9 + m\_Image2.m\_pBits[1][i - 1][j] / 9 + m\_Image2.m\_pBits[1][i][j] / 9 + m\_Image2.m\_pBits[1][i + 1][j] / 9 + m\_Image2.m\_pBits[1][i - 1][j + 1] / 9 + m\_Image2.m\_pBits[1][i][j + 1] / 9 + m\_Image2.m\_pBits[1][i + 1][j + 1] / 9;  m\_Image.m\_pBits[2][i][j] = m\_Image2.m\_pBits[2][i - 1][j - 1] / 9 + m\_Image2.m\_pBits[2][i][j - 1] / 9 + m\_Image2.m\_pBits[2][i + 1][j - 1] / 9 + m\_Image2.m\_pBits[2][i - 1][j] / 9 + m\_Image2.m\_pBits[2][i][j] / 9 + m\_Image2.m\_pBits[2][i + 1][j] / 9 + m\_Image2.m\_pBits[2][i - 1][j + 1] / 9 + m\_Image2.m\_pBits[2][i][j + 1] / 9 + m\_Image2.m\_pBits[2][i + 1][j + 1] / 9;  }  }  Invalidate(1);  } |

2、中值滤波

|  |
| --- |
| void CImage\_ProcessingView::OnMidfilter(){  // TODO: 在此添加命令处理程序代码  if (m\_Image.IsNull()) return;//判断图像是否为空，对空图像操作会出现未知的错误  int w = m\_Image.GetWidth();//获得图像的宽度  int h = m\_Image.GetHeight();//获得图像的高度  int bits = m\_Image.GetBPP();  int hierarchical = 2;  MyImage\_ m\_Image2;  m\_Image2.Load(filename);  for (int i = 1; i < h - 1; i++){  for (int j = 1; j < w - 1; j++){  for (int k = 0; k < 3; k++){  int arr[9] = {m\_Image2.m\_pBits[k][i - 1][j - 1], m\_Image2.m\_pBits[k][i][j - 1], m\_Image2.m\_pBits[k][i + 1][j - 1], m\_Image2.m\_pBits[k][i - 1][j], m\_Image2.m\_pBits[k][i][j], m\_Image2.m\_pBits[k][i + 1][j], m\_Image2.m\_pBits[k][i - 1][j + 1], m\_Image2.m\_pBits[k][i][j + 1], m\_Image2.m\_pBits[k][i + 1][j + 1]}; //将像素点领域内的像素值放到数组内  sort(arr, arr + 9); //对图像像素值进行排序  m\_Image.m\_pBits[k][i][j] = arr[4]; //取排序后的中值  }  }  }  Invalidate(1);  } |

##### 实现结果展示

1. 均值滤波

|  |  |
| --- | --- |
|  |  |

原图 均值滤波后的图

1. 中值滤波

|  |  |
| --- | --- |
|  |  |

原图 中值滤波后的图

### 第四章

编程1：编写程序统计任意一幅图像的直方图并显示

##### 流程图

##### 源代码及重点语句的注释

##### 实现结果展示