

【数字图像处理】五.MFC图像点运算之灰度线性变化、灰度非线性变化、阈值化和均衡化处理详解

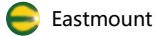
原创 Eastmount 最后发布于2015-06-02 16:04:33 阅读数 13511 ☆ 收藏

展开



Python+TensorFlow人工智能

该专栏为人工智能入门专栏，采用Python3和TensorFlow实现人工智能相关算法。前期介绍安装流程、基础语法...



¥9.90

去订阅

本文主要讲述基于VC++6.0 MFC图像处理的应用知识，主要结合自己大三所学课程《数字图像处理》及课件进行讲解，主要通过MFC单文档视图实现显示BMP图片点运算处理，包括图像灰度线性变换、灰度非线性变换、图像阈值化处理、图像均衡化处理等知识，并结合前一篇论文灰度直方图进行展示。同时文章比较详细基础，希望该篇文章对你有所帮助，尤其是初学者和学习图像处理的学生。

【数字图像处理】一.MFC详解显示BMP格式图片

【数字图像处理】二.MFC单文档分割窗口显示图片

【数字图像处理】三.MFC实现图像灰度、采样和量化功能详解

【数字图像处理】四.MFC对话框绘制灰度直方图

免费资源下载地址：

<http://download.csdn.net/detail/eastmount/8764373>

一. 点运算与初始操作

图像的点运算是图像处理中非常基础的技术，它主要用于改变一幅图像的灰度分布范围，通过一定的变换函数将图像的像素进行转换，最终生成一幅新的图像。点运算的最大特点就是输出像素值只与当前输入像素值相关。定义如下。

点运算(Point Operation)指对于一幅输入图像，将产生一幅输出图像，输出图像的每个像素点的灰度值由输入像素点决定。

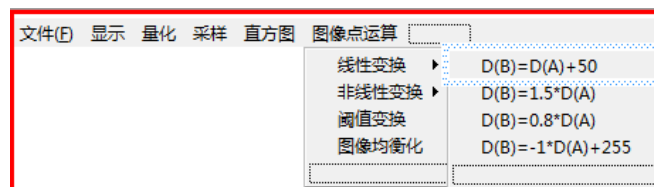
点运算由灰度变换函数(Grayscale Transformation, GST)确定： $B(x,y)=F[A(x,y)]$

需要注意以下几点：

- (1).与局部或邻域运算的差别，输入像素和输出像素是一一对应的；
- (2).与几何运算的差别，不改变图像的空间关系；
- (3).又称为对比增强，对比拉伸或灰度变换。

在前面第四篇博客的基础上增加点运算处理。

第一步：在资源视图中Menu中添加“图像点运算”菜单栏如下所示：



对应的ID值为：

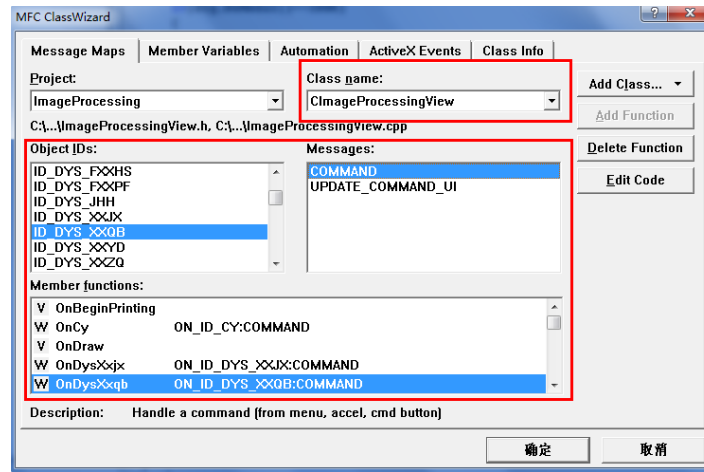
线性变换 ID_DYS_XXYD(点运算 线性移动) ID_DYS_XXZQ(点运算 线性增强)

ID_DYS_XXJX(点运算 线性减小) ID_DYS_XXQB(点运算 线性求补)

非线性变换 ID_DYS_FXXPF(点运算 非线性平方) ID_DYS_FXXHS(非线性函数)

阈值变换 ID_DYS_YZBH(点运算 阈值变换) 图像均衡化 ID_DYS_JHH

第二步：打开类向导(Ctrl+W)，为点运算每个ID菜单添加相应的功能处理函数，如下图所示：选择类CImageProcessingView，在选择IDs为ID_DYS_...(点运算)添加函数OnDysXxqb()线性求补。



二. 线性变换

图像线性变换是通过建立灰度映射来调整资源图像的灰度，从而达到图像增强的目的。其中GST函数 $f(D)$ 为线性的，即：

$$D_B = f(D_A) = \alpha D_A + b$$

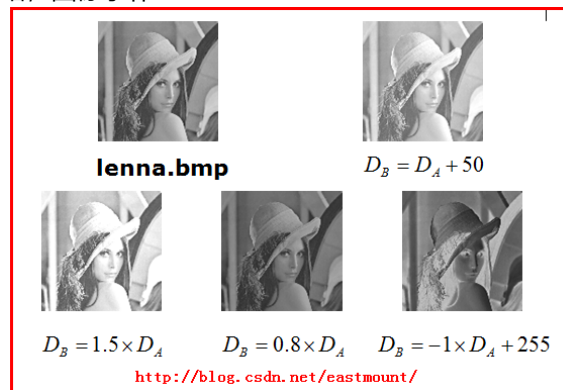
若 $a=1, b=0$ 图像像素不发生变化

若 $a=1, b \neq 0$ 图像所有灰度值上移或下移

若 $a > 1$ 输出图像对比度增强

若 $0 < a < 1$ 输出图像对比度减小

若 $a < 0$ 暗区域变亮，亮区域变暗，图像求补



1. $D(B) = D(A) + 50$

首先是图像移动，代码如下：

```

/*****
/* 图像点运算 4种线性变化直方图：
/* ID_DYS_XXZQ: 表示线性灰度变化移动  $D(B) = D(A) + 50$  灰度值上移下移
/* ID_DYS_XXZQ: 表示线性灰度变化增强  $D(B) = 1.5 * D(A)$  图像对比度增强
/* ID_DYS_XXJX: 表示线性灰度变化减小  $D(B) = 0.8 * D(A)$  图像对比度减小
/* ID_DYS_XXQB: 表示线性灰度求补  $D(B) = -1 * D(A) + 255$  图像暗区变亮，亮区变暗
*****/

```

```

// 1. 点运算 线性灰度变化移动  $D(B) = D(A) + 50$ 

```

```

void CImageProcessingView::OnDysXxyd()
{

```

```

    // TODO: Add your command handler code here

```

```

    if(numPicture==0) {

```

```

        AfxMessageBox("载入图片后才能线性灰度运算!", MB_OK, 0);
    }
}

```

```

        return;    }
AfxMessageBox("线性灰度直方图-灰度变化移动 D(B)=D(A)+50!",MB_OK,0);
int i;
// 打开临时的图片
FILE *fpo = fopen(BmpName,"rb");
FILE *fpw = fopen(BmpNameLin,"wb+");
// 读取文件
fread(&bfh,sizeof(BITMAPFILEHEADER),1,fpo);
fread(&bih,sizeof(BITMAPINFOHEADER),1,fpo);
fwrite(&bfh,sizeof(BITMAPFILEHEADER),1,fpw);
fwrite(&bih,sizeof(BITMAPINFOHEADER),1,fpw);
// 灰度图像
unsigned char color;
unsigned char red,green,blue;
for( i=0; i<m_nImage/3; i++ )
{
    fread(&red,sizeof(char),1,fpo);
    fread(&green,sizeof(char),1,fpo);
    fread(&blue,sizeof(char),1,fpo);

    if( (int)red+50 >255 )
        red=255;
    else
        red=(int)red+50;

    if( (int)green+50>255 )
        green=255;
    else
        green=(int)green+50;

    if( (int)blue+50>255 )
        blue=255;
    else
        blue=(int)blue+50;

    fwrite(&red,sizeof(char),1,fpw);
    fwrite(&green,sizeof(char),1,fpw);
    fwrite(&blue,sizeof(char),1,fpw);
}
fclose(fpo);
fclose(fpw);
numPicture = 2;
level=101;    // 赋值101在ShowBitmap中调用显示处理后的图片
Invalidate();
}

```

同时修改void CImageProcessingView::ShowBitmap(CDC *pDC, CString BmpName)函数中的代码:

```

else    // 图像点运算 线性变化
if(level=101)
{
    m_hBitmapChange = (HBITMAP) LoadImage(NULL,BmpNameLin,IMAGE_BITMAP,0,0,
        LR_LOADFROMFILE|LR_DEFAULTSIZE|LR_CREATEDIBSECTION);
}

```

运行效果如下图所示, 同时我截取了直方图(RGB相同只显示一种)。



可以发现图像的灰度上移了50，图像更白了（黑0-255白）。

2.D(B)=1.5*D(A)

// 2.点运算 线性灰度变化增强 $D(B)=1.5*D(A)$

void CImageProcessingView::OnDysXxzq()

```
{
    if(numPicture==0) {
        AfxMessageBox("载入图片后才能线性灰度运算!",MB_OK,0);
        return;
    }
    AfxMessageBox("线性灰度直方图-灰度变化增强  $D(B)=1.5*D(A)$ !",MB_OK,0);
    int i;
    // 打开临时的图片
    FILE *fpo = fopen(BmpName,"rb");
    FILE *fpw = fopen(BmpNameLin,"wb+");
    fread(&bfh,sizeof(BITMAPFILEHEADER),1,fpo);
    fread(&bih,sizeof(BITMAPINFOHEADER),1,fpo);
    fwrite(&bfh,sizeof(BITMAPFILEHEADER),1,fpw);
    fwrite(&bih,sizeof(BITMAPINFOHEADER),1,fpw);
    // 灰度图像
    unsigned char color;
    unsigned char red,green,blue;
    for( i=0; i<m_nImage/3; i++ )
    {
        fread(&red,sizeof(char),1,fpo);
        fread(&green,sizeof(char),1,fpo);
        fread(&blue,sizeof(char),1,fpo);

        if( (int)red*1.5 >255 )
            red=255;
        else
            red=(int)red*1.5;

        if( (int)green*1.5>255 )
            green=255;
        else
            green=(int)green*1.5;

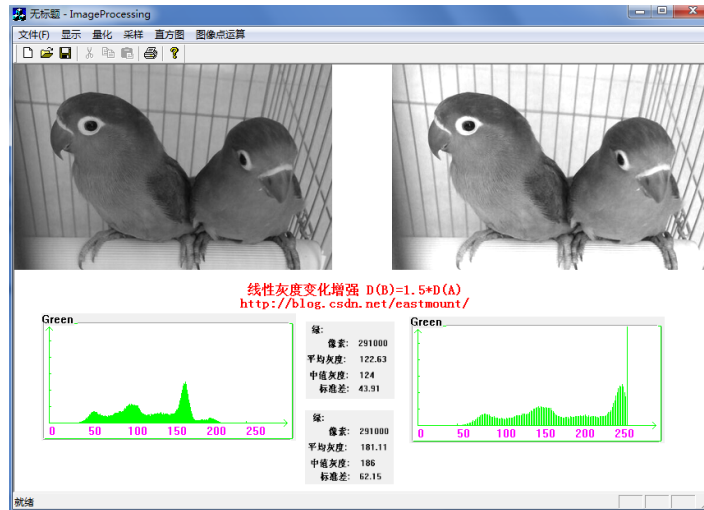
        if( (int)blue*1.5>255 )
            blue=255;
        else
            blue=(int)blue*1.5;
    }
}
```

```

        fwrite(&red,sizeof(char),1,fpw);
        fwrite(&green,sizeof(char),1,fpw);
        fwrite(&blue,sizeof(char),1,fpw);
    }
    fclose(fpo);
    fclose(fpw);
    numPicture = 2;
    level=101;    //线性变化 ShowBitmap中调用
    Invalidate();
}

```

运行效果如下图所示，图像对比度增强，平均灰度 $122 \times 1.5 = 181$



3. $D(B)=0.8 \times D(A)$

// 3. 点运算 线性灰度变化减小 $D(B)=0.8 \times D(A)$

```

void CImageProcessingView::OnDysXxjx()
{
    if(numPicture==0) {
        AfxMessageBox("载入图片后才能线性灰度处理!",MB_OK,0);
        return;
    }
    AfxMessageBox("线性灰度直方图-灰度减小  $D(B)=0.8 \times D(A)$ !",MB_OK,0);
    int i;
    // 打开临时的图片
    FILE *fpo = fopen(BmpName,"rb");
    FILE *fpw = fopen(BmpNameLin,"wb+");
    fread(&bfh,sizeof(BITMAPFILEHEADER),1,fpo);
    fread(&bih,sizeof(BITMAPINFOHEADER),1,fpo);
    fwrite(&bfh,sizeof(BITMAPFILEHEADER),1,fpw);
    fwrite(&bih,sizeof(BITMAPINFOHEADER),1,fpw);
    // 灰度图像
    unsigned char color;
    unsigned char red,green,blue;
    for( i=0; i<m_nImage/3; i++ )
    {
        fread(&red,sizeof(char),1,fpo);
        fread(&green,sizeof(char),1,fpo);
        fread(&blue,sizeof(char),1,fpo);

        red=(int)red*0.8;
        green=(int)green*0.8;
        blue=(int)blue*0.8;
    }
}

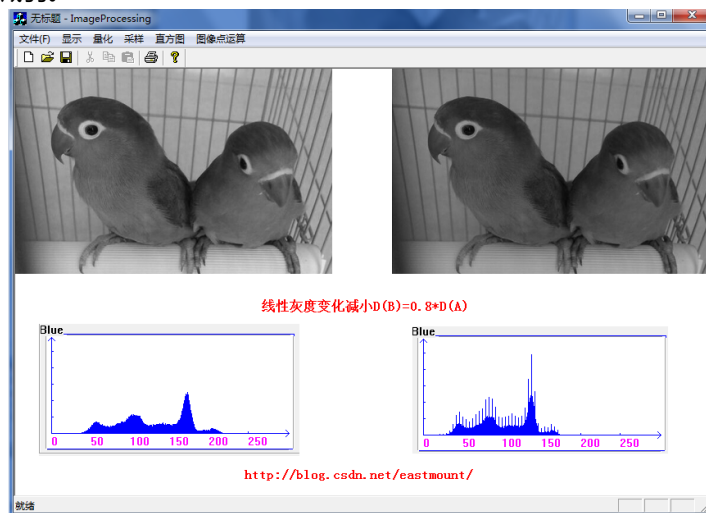
```

```

        fwrite(&red,sizeof(char),1,fpw);
        fwrite(&green,sizeof(char),1,fpw);
        fwrite(&blue,sizeof(char),1,fpw);
    }
    fclose(fpo);
    fclose(fpw);
    numPicture = 2;
    level=101;
    Invalidate();
}

```

运行如下图所示，图像减弱。



4.D(B)=-1*D(A)+255

// 4. 点运算 线性灰度求补 $D(B) = -1 * D(A) + 255$

```

void CImageProcessingView::OnDysXxb()
{
    if(numPicture==0) {
        AfxMessageBox("载入图片后才能线性灰度处理!",MB_OK,0);
        return;
    }
    AfxMessageBox("线性灰度直方图-灰度求补  $D(B) = -1 * D(A) + 255$ !",MB_OK,0);
    int i;
    // 打开临时的图片
    FILE *fpo = fopen(BmpName,"rb");
    FILE *fpw = fopen(BmpNameLin,"wb+");
    fread(&bfh,sizeof(BITMAPFILEHEADER),1,fpo);
    fread(&bih,sizeof(BITMAPINFOHEADER),1,fpo);
    fwrite(&bfh,sizeof(BITMAPFILEHEADER),1,fpw);
    fwrite(&bih,sizeof(BITMAPINFOHEADER),1,fpw);
    // 灰度图像
    unsigned char color;
    unsigned char red,green,blue;
    for( i=0; i<m_nImage/3; i++ )
    {
        fread(&red,sizeof(char),1,fpo);
        fread(&green,sizeof(char),1,fpo);
        fread(&blue,sizeof(char),1,fpo);

        red=(int)red*(-1)+255;
        green=(int)green*(-1)+255;

```

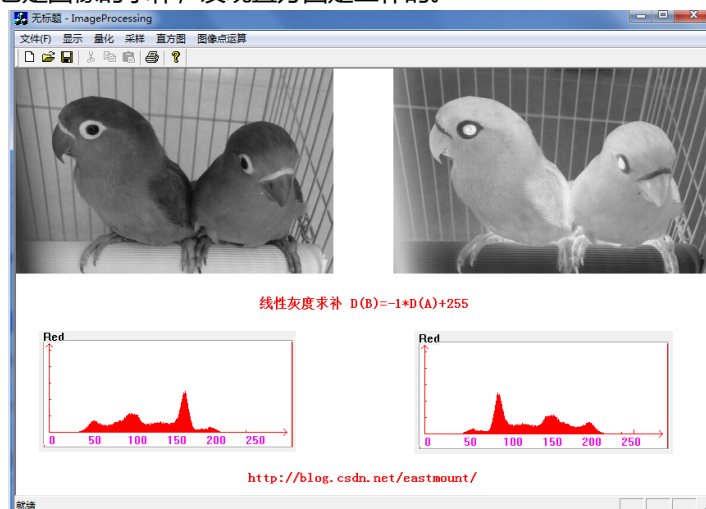
```

        blue=(int)blue*(-1)+255;

        fwrite(&red,sizeof(char),1,fpw);
        fwrite(&green,sizeof(char),1,fpw);
        fwrite(&blue,sizeof(char),1,fpw);
    }
    fclose(fpo);
    fclose(fpw);
    numPicture = 2;
    level=101;
    Invalidate();
}

```

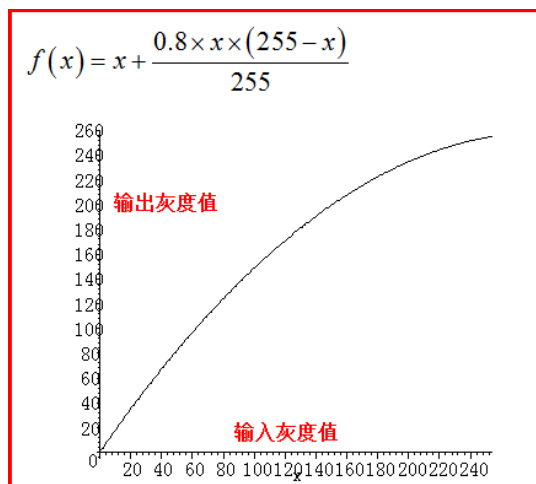
运行效果如下图所示，它是图像的求补，发现直方图是互补的。



PS:注意图片下面的直方图应该还有一个处理后的直方图，但原理都一样，我不想重复工作，你自己可以去简单实现下，参考第四篇文章。同时这些图片制作还挺麻烦的，只是为了给你更好的呈现它们的变化，希望对你有用和尊重作者，不喜勿喷~

三. 非线性变换

灰度非线性变换主要包括对数变换、幂次变换、指数变换、分段函数变换，通过非线性关系对图像进行灰度处理，下面主要讲解课件中的两个函数对其进行处理。其中对数变换实现了扩展低灰度值而压缩高灰度值的效果，图像灰度分布更符合而你的视觉特征。



$$1.D(B)=D(A)*D(A)/252$$

```

/*****
/* 2种非线性变化直方图:
/* ID_DYS_FXXPF:表示非线性平方灰度变化, $D(B)=D(A)*D(A)/255$ 
/* ID_DYS_FXXHS:表示非线性函数灰度变化, $D(B)=D(A)+0.8*D(A)*(255-D(A))/255$ 
*****/

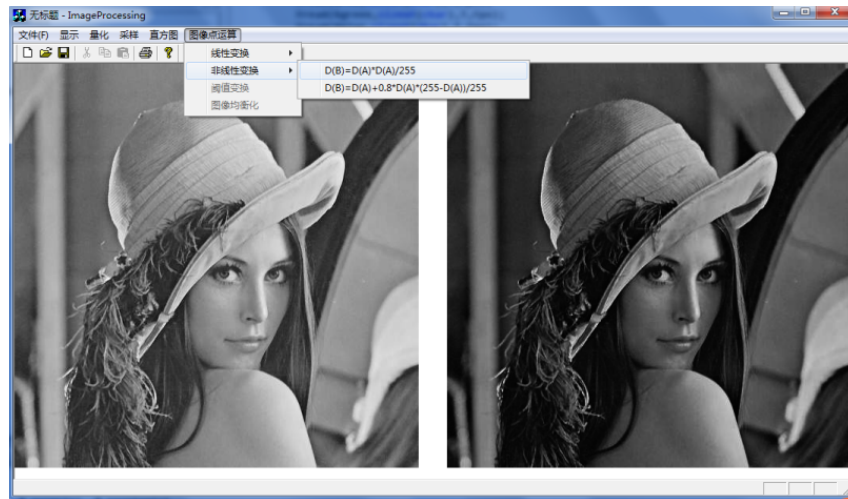
// 非线性平方灰度变化  $D(B)=D(A)*D(A)/252$ 
void CImageProcessingView::OnDysFxxpf()
{
    if(numPicture==0)
    {
        AfxMessageBox("载入图片后才能非线性灰度处理!",MB_OK,0);
        return;
    }
    AfxMessageBox("非线性灰度变化  $D(B)=D(A)*D(A)/255!$ ",MB_OK,0);
    int i;
    // 打开临时的图片
    FILE *fpo = fopen(BmpName,"rb");
    FILE *fpw = fopen(BmpNameLin,"wb+");
    // 读取文件
    fread(&bfh,sizeof(BITMAPFILEHEADER),1,fpo);
    fread(&bih,sizeof(BITMAPINFOHEADER),1,fpo);
    fwrite(&bfh,sizeof(BITMAPFILEHEADER),1,fpw);
    fwrite(&bih,sizeof(BITMAPINFOHEADER),1,fpw);
    // 灰度图像
    unsigned char color;
    unsigned char red,green,blue;
    for( i=0; i<m_nImage/3; i++ )
    {
        fread(&red,sizeof(char),1,fpo);
        fread(&green,sizeof(char),1,fpo);
        fread(&blue,sizeof(char),1,fpo);

        red=(int)red*(int)red/255;
        green=(int)green*(int)green/255;
        blue=(int)blue*(int)blue/255;

        fwrite(&red,sizeof(char),1,fpw);
        fwrite(&green,sizeof(char),1,fpw);
        fwrite(&blue,sizeof(char),1,fpw);
    }
    fclose(fpo);
    fclose(fpw);
    numPicture = 2;
    level=101;
    Invalidate();
}

```

运行效果如下图所示:



2. $D(B)=D(A)+0.8*D(A)*(255-D(A))/255$

// 非线性函数灰度变化 $D(B)=D(A)+0.8*D(A)*(255-D(A))/255$

```
void CImageProcessingView::OnDysFxxhs()
{
    if(numPicture==0)
    {
        AfxMessageBox("载入图片后才能非线性灰度处理!", MB_OK, 0);
        return;
    }
    AfxMessageBox("线性灰度直方图-灰度变化增强  $D(B)=D(A)+0.8*D(A)*(255-D(A))/255$ !", MB_OK, 0);
    int i;

    FILE *fpo = fopen(BmpName, "rb");
    FILE *fpw = fopen(BmpNameLin, "wb+");
    fread(&bfh, sizeof(BITMAPFILEHEADER), 1, fpo);
    fread(&bih, sizeof(BITMAPINFOHEADER), 1, fpo);
    fwrite(&bfh, sizeof(BITMAPFILEHEADER), 1, fpw);
    fwrite(&bih, sizeof(BITMAPINFOHEADER), 1, fpw);

    unsigned char color;
    unsigned char red, green, blue;
    for( i=0; i<m_nImage/3; i++ )
    {
        fread(&red, sizeof(char), 1, fpo);
        fread(&green, sizeof(char), 1, fpo);
        fread(&blue, sizeof(char), 1, fpo);

        if( ((int)red+0.8*(int)red*(255-(int)red)/255) > 255 )
            red=255;
        else
            red=(int)red+0.8*(int)red*(255-(int)red)/255;

        if( ((int)green+0.8*(int)green*(255-(int)green)/255) > 255 )
            green=255;
        else
            green=(int)green+0.8*(int)green*(255-(int)green)/255;

        if( ((int)blue+0.8*(int)blue*(255-(int)blue)/255) > 255 )
            blue=255;
        else
            blue=(int)blue+0.8*(int)blue*(255-(int)blue)/255;

        fwrite(&red, sizeof(char), 1, fpw);
    }
}
```

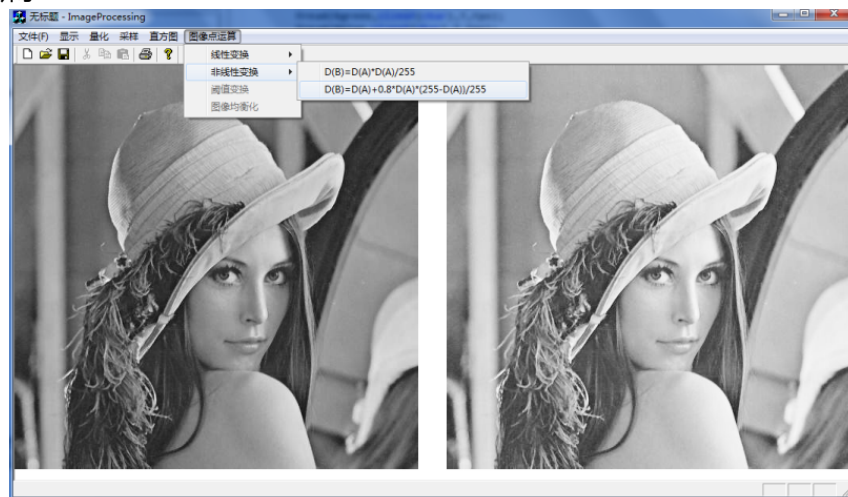
```

        fwrite(&green,sizeof(char),1,fpw);

        fwrite(&blue,sizeof(char),1,fpw);
    }
    fclose(fpo);
    fclose(fpw);
    numPicture = 2;
    level=101;
    Invalidate();
}

```

运行效果如下图所示:



写到此处你会发现图像灰度的线性变换和非线性变换是非常简单的，主要是通过以下步骤完成：

第一步：赋值处理后图像的BMP头信息

```

FILE *fpo = fopen(BmpName,"rb");
FILE *fpw = fopen(BmpNameLin,"wb+");
fread(&bfh,sizeof(BITMAPFILEHEADER),1,fpo);
fread(&bih,sizeof(BITMAPINFOHEADER),1,fpo);
fwrite(&bfh,sizeof(BITMAPFILEHEADER),1,fpw);
fwrite(&bih,sizeof(BITMAPINFOHEADER),1,fpw);

```

第二步：通过循环和线性变换或非线性变换函数处理每一个像素

```

for( i=0; i<m_nImage/3; i++ )
{
    fread(&red,sizeof(char),1,fpo);
    处理像素RGB 如:red=(int)red*(int)red/255;
    fwrite(&red,sizeof(char),1,fpw);
}

```

第三步：调用ShowBitmap自定义函数并重绘图像

```

numPicture = 2;
level=101;
Invalidate();

```

而它的主要应用包括：光度学标定，希望数字图像的灰度能够真实反映图像的物理特性；对比度增强和对比度扩展；显示标定和轮廓线确定(阈值化)。

四. 灰度阈值化

阈值又称为临界值，它的目的是确定出一个范围，然后这个范围内的部分使用同一种方法处理，而阈值之外的部分则

使用另一种处理方法或保持原样。常用的包括产生二值图：当 $x < T$ 时 $y = 0$ ，当 $x \geq T$ 时 $y = 255$ （其中 T 是阈值）。阈值变换在生物学上的应用比较广泛，常用语细胞图像分割等。

打开类向导(Ctrl+W)生成选择ImageProcessingView类，IDs选择ID_DYS_YZBH后添加相应的函数。代码如下：

```

/*****
/* ID_DYS_YZBH: 表示点运算阈值变换 也看做灰度拉伸
/* 此处的拉伸是: 阈值化(thresholding)可以看作是削波的一个特例
/* 只要令削波中的g1old=g2old就实现了阈值化。
/* 阈值就象个门槛, 比它大就是白, 比它小就是黑, 二值
*****/

void CImageProcessingView::OnDysYzbh()
{
    if(numPicture==0)
    {
        AfxMessageBox("载入图片后才能点运算阈值化处理!",MB_OK,0);
        return;
    }
    AfxMessageBox("图像点运算阈值化处理!",MB_OK,0);
    // 读写文件
    FILE *fpo = fopen(BmpName,"rb");
    FILE *fpw = fopen(BmpNameLin,"wb+");
    fread(&bfh,sizeof(BITMAPFILEHEADER),1,fpo);
    fread(&bih,sizeof(BITMAPINFOHEADER),1,fpo);
    fwrite(&bfh,sizeof(BITMAPFILEHEADER),1,fpw);
    fwrite(&bih,sizeof(BITMAPINFOHEADER),1,fpw);
    // 处理
    unsigned char color;
    unsigned char red,green,blue;
    for(int i=0; i<m_nImage/3; i++ )
    {
        fread(&red,sizeof(char),1,fpo);
        fread(&green,sizeof(char),1,fpo);
        fread(&blue,sizeof(char),1,fpo);

        if( (int)red > 128 )
            red=255;
        else
            red=0;

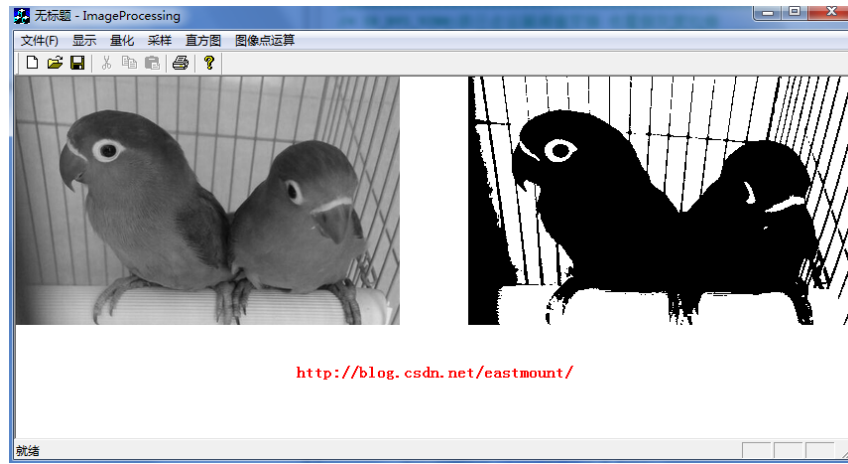
        if( (int)green > 128 )
            green=255;
        else
            green=0;

        if( (int)blue > 128 )
            blue=255;
        else
            blue=0;

        fwrite(&red,sizeof(char),1,fpw);
        fwrite(&green,sizeof(char),1,fpw);
        fwrite(&blue,sizeof(char),1,fpw);
    }
    fclose(fpo);
    fclose(fpw);
    numPicture = 2;
    level=101;
    Invalidate();
}

```

运行效果如下图所示, 感觉还挺好看的, 显然此时的直方图就是0和255两条直线。



五. 灰度均衡化

灰度均衡化的目的是使一输入图像转换为在每一灰度级上都有相同的像素点（即输出的直方图是平的），它可以产生一幅灰度级分布概率均衡的图像。

换句话说，经过均衡化后的图像在每一级灰度上像素点的数量相差不大，对应的灰度直方图的每一级高度也相差不大。它是增强图像的有效手段之一。

研究思路是通过直方图变换公式实现：

$$H_B(D) = \frac{H_A[f^{-1}(D)]}{f'[f^{-1}(D)]}$$

它的步骤如下图所示：

$$\text{step1: } \frac{A_0}{D_m} = \frac{H_A[f^{-1}(D)]}{f'[f^{-1}(D)]}$$

$$\text{step2: } f' = \frac{D_m}{A_0} H(D)$$

$$\text{step3: } f(D) = \frac{D_m}{A_0} \int_0^D H(u) du$$

$$\text{step4: } \therefore CDF(D) = \frac{1}{A_0} \int_0^D H(u) du$$

$$\text{step5: } \therefore f(D) = D_m \times CDF(D)$$

例：有一幅图象，共有16级灰度，其直方图分布为 P_i , $i=0,1,\dots,15$ ，求经直方图均衡化后，量化级别为10级的灰度图象的直方图分布 Q_i ，其中 P_i 和 Q_i 为分布的概率，即灰度 i 出现的次数与总的点数之比。

P_i : 0.03, 0, 0.06, 0.10, 0.20, 0.11, 0, 0, 0, 0.03, 0, 0.06, 0.10, 0.20, 0.11, 0

步骤1：用一个数组 s 记录 P_i ，即 $s[0]=0.03, s[1]=0, s[2]=0.06, \dots, s[14]=0.11, s[15]=0$

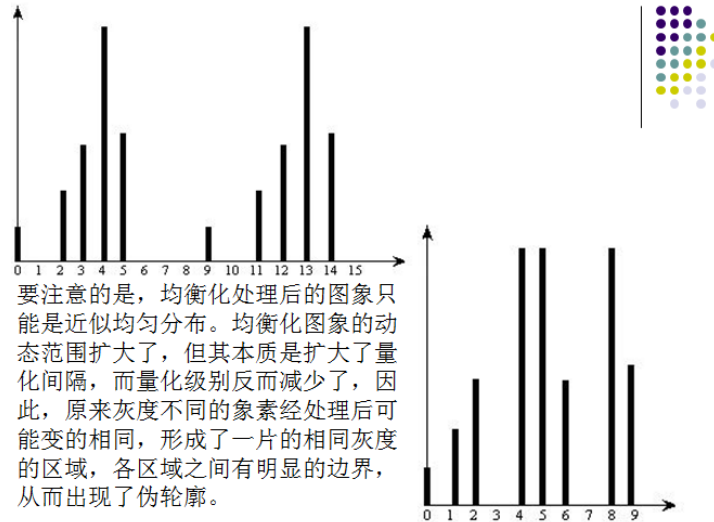
步骤2： i 从1开始，令 $s[i]=s[i]+s[i-1]$ ，得到的结果是 s : 0.03, 0.03, 0.09, 0.19, 0.39, 0.50, 0.50, 0.50, 0.50, 0.53, 0.53, 0.59, 0.69, 0.89, 1.0, 1.0

步骤3：用一个数组 L 记录新的调色板索引值，即令 $L[i]=s[i] \times (10-1)$ ，得到的结果是 L : 0,0,1,2,4,5,5,5,5,5,5,6,8,9,9
这样就找到了原来的调色板索引值和新的调色板索引值之间的对应关系，即

0→0, 1→0, 2→1, 3→2, 4→4, 5→5, 6→5, 7→5, 8→5, 9→5, 10→5, 11→5, 12→6, 13→8, 14→9, 15→9。

步骤4：将老的索引值对应的概率合并，作为对应的新的索引值的概率。例如，原来的索引值0,1都对应了新的索引值

0, 则灰度索引值为0的概率为 $P_0+P_1=0.03$; 新的索引值3和7找不到老的索引值与之对应, 所以令 Q_3 和 Q_7 为0。最后得到的结果是 Q_i : 0.03, 0.06, 0.10, 0, 0.20, 0.20, 0.10, 0, 0.20, 0.11



代码中有详细注释如下:

```
// ID_DYS_JHH: 表示图像均衡化 相称算法
void CImageProcessingView::OnDysJhh()
{
    if(numPicture==0) {
        AfxMessageBox("载入图片后才能图像均衡化!", MB_OK, 0);
        return;
    }
    AfxMessageBox("图像均衡化!", MB_OK, 0);

    // 第一步: 获取图像的数据信息
    // 此操作可以在打开图片时就进行 在直方图采样(ZFTCY)中也有该代码
    FILE *fpo = fopen(BmpName, "rb");
    fread(&bfh, sizeof(BITMAPFILEHEADER), 1, fpo);
    fread(&bih, sizeof(BITMAPINFOHEADER), 1, fpo);

    int i, j, k;
    for(j=0; j<256; j++) { // 定义数组并清零
        Red[j]=0;
        Green[j]=0;
        Blue[j]=0;
    }

    // 计算4个数据
    unsigned char red, green, blue;
    int IntRed, IntGreen, IntBlue; // 强制转换
    double sumRedHD=0, sumGreenHD=0, sumBlueHD=0; // 记录像素总的灰度值和
    for(i=0; i<m_nImage/3; i++)
    {
        fread(&red, sizeof(char), 1, fpo);
        IntRed=int(red);
        sumRedHD=sumRedHD+IntRed;
        if( IntRed>=0 && IntRed<256 ) Red[IntRed]++;

        fread(&green, sizeof(char), 1, fpo);
        IntGreen=int(green);
        sumGreenHD=sumGreenHD+IntGreen;
        if( IntGreen>=0 && IntGreen<256 ) Green[IntGreen]++;

        fread(&blue, sizeof(char), 1, fpo);
```

```

        IntBlue=int(blue);
                                sumBlueHD=sumBlueHD+IntBlue;
        if( IntBlue>=0 && IntBlue<256 ) Blue[IntBlue]++;
    }
    fclose(fpo);

    /*****
    /* 图像均衡化处理
    /* 利用全局变量 Red[256] Blue[256] Green[256]
    /* 第一步:用3个数组Count..记录0-255灰度出现的概率,即
    /*      概率=该灰度出现次数*3/总得像素 (因为分成3部分RGB)
    /* 第二步:i从1开始,令s[i]=s[i]+s[i-1] 记录新概率数
    /* 第三步:用一个数组L记录新的调色板索引值,即
    /*      L[i]=s[i]*(256-1)结果四舍五入2.8即为3
    /* 第四步:将老的索引值对应的概率合并,作为对应的新的索引值的概率
    /*      1.原来的索引值0,1都对应了新的索引值0,则灰度索引值为0的概率
    /*          为P0+P1=0.03
    /*      2.新的索引值3和7找不到老的索引值与之对应,所以令Q3和Q7为0
    *****/

    //记录出现的概率,会加到1 用于相加到调色板
    float CountRed[256],CountGreen[256],CountBlue[256];
    //记录原始数据,不会相加到1 用于计算新灰度概率
    float CountRedLin[256],CountGreenLin[256],CountBlueLin[256];

    for( k=0 ; k<256 ; k++ )
    {
        CountRed[k]=(float)(Red[k])*3/m_nImage;
        CountRedLin[k]=CountRed[k];
        CountGreen[k]=(float)(Green[k])*3/m_nImage;
        CountGreenLin[k]=CountGreen[k];
        CountBlue[k]=(float)(Blue[k])*3/m_nImage;
        CountBlueLin[k]=CountBlue[k];
    }

    for( k=1 ; k<256 ; k++ )
    {
        CountRed[k]=CountRed[k]+CountRed[k-1];
        CountGreen[k]=CountGreen[k]+CountGreen[k-1];
        CountBlue[k]=CountBlue[k]+CountBlue[k-1];
    }

    /*****
    /* 此处百度到一个四舍五入浮点型的算法:
    /* float a=3.456; 保留到小数点后两位
    /* float b=(int)((a * 100) + 0.5) / 100.0;
    /* output b=3.46
    *****/

    int LRed[256],LGreen[256],LBlue[256]; //记录调色板
    for( k=0 ; k<256 ; k++ )
    {
        LRed[k]=(int)(CountRed[k]*(256-1)+0.5);
        LGreen[k]=(int)(CountGreen[k]*(256-1)+0.5);
        LBlue[k]=(int)(CountBlue[k]*(256-1)+0.5);
    }

    //第三步:处理均衡化图像写入 打开临时的图片
    fpo = fopen(BmpName,"rb");
    fread(&bhf,sizeof(BITMAPFILEHEADER),1,fpo);
    fread(&bih,sizeof(BITMAPINFOHEADER),1,fpo);

```

```

FILE *fpw = fopen(BmpNameLin,"wb+");
fwrite(&bfh,sizeof(BITMAPFILEHEADER),1,fpw);
fwrite(&bih,sizeof(BITMAPINFOHEADER),1,fpw);

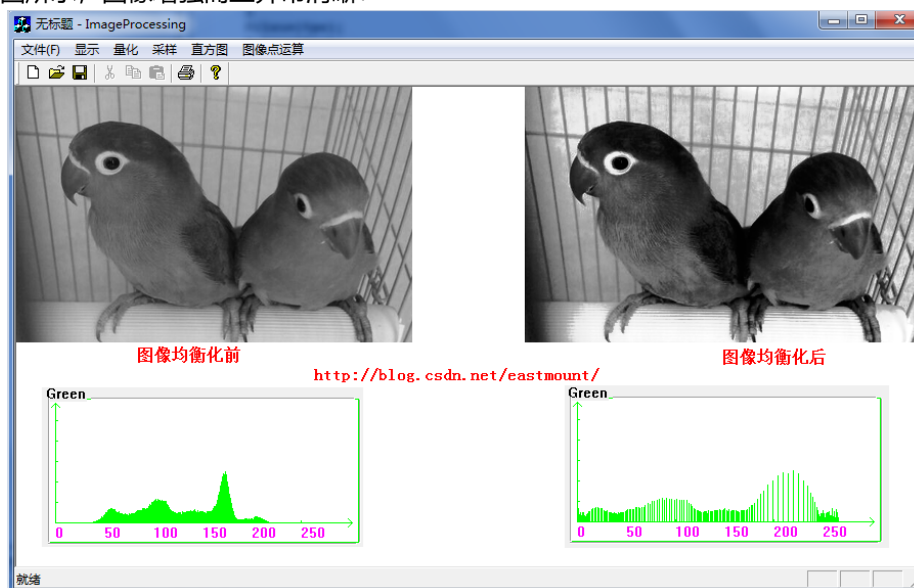
//m_nWidth*m_nHeight 读取图片最后一行不为m_nWidth时会报错 改为m_nImage/3
for( i=0; i<m_nImage/3 ; i++ )
{
    fread(&red,sizeof(char),1,fpo);
    fread(&green,sizeof(char),1,fpo);
    fread(&blue,sizeof(char),1,fpo);

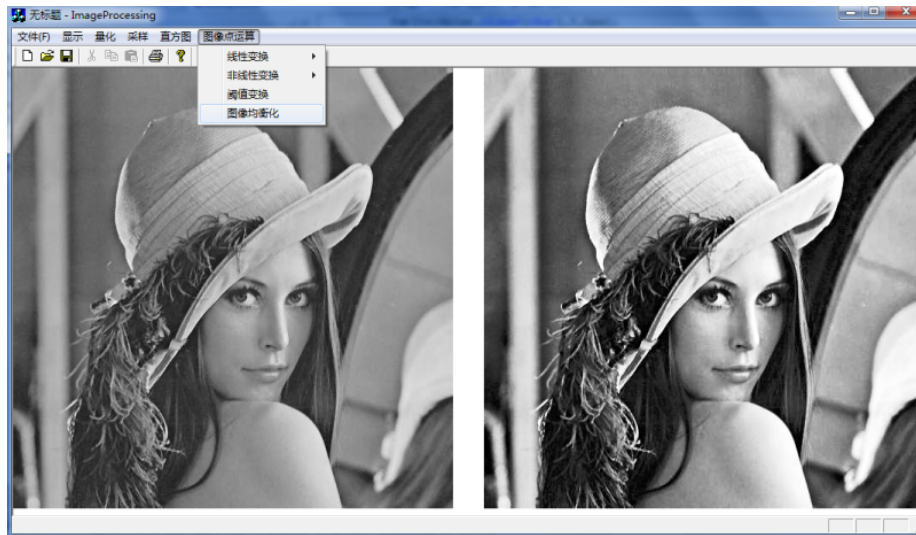
    red=LRed[int(red)];
    green=LGreen[int(green)];
    blue=LBlue[int(blue)];

    fwrite(&red,sizeof(char),1,fpw);
    fwrite(&green,sizeof(char),1,fpw);
    fwrite(&blue,sizeof(char),1,fpw);
}
fclose(fpw);
numPicture = 2;
level=101;
Invalidate();
}

```

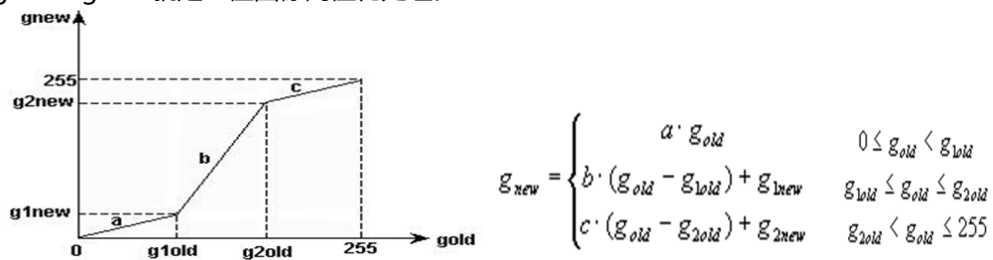
运行结果如下图所示，图像增强而且异常清晰：





最后介绍下图像对比度拉伸，它就是把你感兴趣的灰度范围拉开，使得该范围内像素，亮的更亮，暗的更暗，从而达到增强对比度的目的。

如下图所示，a、b、c为三段直线的斜率，g1old和g2old表示途中要进行对比度扩展的范围，g1new和g2new表示对应的新值。当g1old=g2old就是二值图像阈值化处理。



由于灰度界别也是255这个约束，所以满足

$$ag_{1old} + b(g_{2old} - g_{1old}) + c(255 - g_{2old}) = 255$$

其中g1old=100, g2old=150, b=3.0的运行效果如下所示：




最后还是希望文章对你有所帮助，如果文章有不足或错误之处，请海涵。最近挺忙的，写这些古老的文章有人说在浪费青春，但我还是准备把这个系列讲完，非常高兴以前的代码注释和风格都不错，回忆起来挺好的，希望你也能养成好的代码和注释风格~

(By:Eastmount 2015-06-02 下午4点 <http://blog.csdn.net/eastmount/>)

👍 点赞 11 ☆ 收藏 🔄 分享 ...



Eastmount  博客专家

发布了450 篇原创文章 · 获赞 6318 · 访问量 501万+

他的留言板

关注