2016-3-31

HOMEWORK5

Cecil Wang
MICROSOFT

目录

一、作业描述

编程实现一种完美直方图均衡化或直方图匹配算法（直方图完全匹配而不是近似匹配），并和正常直方图均衡化或直方图匹配结果作比较。

二、直方图均衡化

直方图拉伸是通过对比度拉伸对直方图进行调整，从而“扩大”前景和背景灰度的差别，以达到增强对比度的目的，通常增加对比度可以利用线性或非线性的方法来实现;而直方图均衡化则通过使用累积函数对灰度值进行“调整”以实现对比度的增强。这种方法通常用来增加许多图像的局部对比度，尤其是当图像的有用数据的对比度相当接近的时候。

直方图均衡化处理的“中心思想”是把原始图像的灰度直方图从比较集中的某个灰度区间变成在全部灰度范围内的均匀分布，它对图像进行非线性拉伸，重新分配图像像素值，使一定灰度范围内的像素数量大致相同，将给定图像的直方图分布改变成“均匀”分布直方图分布。

为了完成直方图均衡化操作，同时直方图匹配也需要的两个功能是计算图像各个通道的直方图，以及计算累计直方图，这两个功能我写了两个函数文件对应，分别是 calculateHist.m

```
% 计算各个通道的直方图
function hist = calculateHist( Img )
channel = size(Img,3);
hist = zeros(256, channel);
for i = 0:255
    for j = 1:channel
        %通过 find 操作获取坐标，然后通过 length 获取个数
        hist(i+1,j) = length(find(Img(:,:,j)==i));
    end
end
end
```

和 calculateCDF.m。

```
% 计算累计直方图
function CDF = calculateCDF( Img )
[H,W,channel] = size(Img);
hist = calculateHist(Img);
%直接通过 cumsum 算出各个通道的累计直方图
CDF = cumsum(hist, 1);
%归一化到【0-1】
CDF = CDF / double(H) / double(W);
CDF(find(CDF>255)) = 255;
CDF(find(CDF<0)) = 0;
end
```

下面我们来关注均衡化操作 equalization.m。

```
function dstImg = equalization(srcImg)
% srcImg:  the source image
% dstImg:  the result image


%首先将灰度图处理成三维的数组
if length(size(srcImg)) == 2
    [H, W] = size(srcImg);
    srcImg = reshape(srcImg, [H,W,1]);
end
[H,W,channel] = size(srcImg);

%获取累计分布函数
CDF = calculateCDF(srcImg);

dstImg = zeros(H,W,channel);
for i = 1:channel
    %每一个 channel 生成一个临时存储图像，直接通过像素值作为索引访问 CDF
    tmp = CDF(srcImg(:,:,i)+1, i);
    dstImg(:,:,i) = reshape(tmp, [H,W]);
end

%将灰度图变成 2 维数组
if channel == 1
   dstImg = reshape(dstImg,[H,W]);
end

end
```

实验结果

三、直方图匹配

3.1 直方图匹配

直方图匹配是指将一幅图像的直方图变成规定形状的直方图而进行的图像增强操作，通常规定的直方图是通过另外一张图片的直方图决定的。其中心思想是首先计算两个直方图的累计分布函数，然后将原图片的直方图找到目的图片直方图最接近的点作为映射关系。具体详见 matching.m。

实验结果

```
function resultImg = matching(srcImg, dstImg)
% srcImg:       the source image
% dstImg:       the destination image
% resultImg:    the result image

%计算累计分布函数
srcCDF = calculateCDF(srcImg);
dstCDF = calculateCDF(dstImg);

%建立 Look-up-table
LUT = zeros(256,channel);
for i = 1:256
    for j = 1:channel
```

```
        %找到比它大的所有索引位置
         tmp = find(dstCDF(:,j)>=srcCDF(i,j));
        %特殊处理没有索引的情况
         if length(tmp)  == 0
             LUT(i,j) = 1.0;
        %选取最小的一个作为映射
         else
             LUT(i,j) = (tmp(1)-1)/255.0;
         end
    end
end

%通过 LUT 映射像素
resultImg = zeros(H,W,channel);
for i = 1:channel
    tmp = LUT(srcImg(:,:,i)+1, i);
    resultImg(:,:,i) = reshape(tmp,[H,W]);
end

end
```

实验结果

3.2 直方图完美匹配

个人认为直方图完美匹配没有什么实际意义，为了保证完美匹配，意味着原图片的直方图和目的图片的直方图完全一样，这种映射存在着很大的问题。首先，当一个目的图片某一个灰度级中的点多于原图片此灰度级中的像素点时，意味着我们需要将原图片中其他灰度级的像素点变成此灰度级，在如何选择像素点上是一个难以抉择的问题。即使我们假设选择与其最近的灰度级来变化，那么在这个灰度级中选择哪一部分的像素点也是一个问题。当问题扩大到彩色图时，问题会变得更加严重，比如如何衡量两种颜色的相似性等。

在我的实验中，我假设每个通道的像素点是相互独立的，然后在最近的灰度级中按照从左上角到右下角的顺序选择需要变化的像素点。具体的，perfectmatch.m。

```
function resultImg = perfectmatch(srcImg, dstImg)
% srcImg:       the source image
% dstImg:       the destination image
% resultImg:    the result image
[srcH,srcW,srcChannel] = size(srcImg);
[dstH,dstW,dstChannel] = size(dstImg);
%为了方便我们将目的图像缩放到原图像大小
dstImg = imresize(dstImg,[srcH, srcW]);

%只需要使用直方图，不需要累积分布函数
srcHist = calculateHist(srcImg);
dstHist = calculateHist(dstImg);
resultImg = zeros(srcH,srcW,srcChannel);
for j = 1:srcChannel
    excessPiexl = [];
    tmpImg=zeros(srcH*srcW, 1);

    %将所有原图片中多余的点取出放到 excessPiexl 备用
    for i = 1:256
        if( srcHist(i,j) > dstHist(i,j))
            val = srcHist(i,j) - dstHist(i,j);
            id = find(srcImg(:,:,j)==i-1);
            excessPiexl = [ excessPiexl id(1:val)' ];
            tmpImg(id(val+1:end)) = i-1;
        end
    end

    %当原图片总像素点不够时，从 excessPiexl 中取出填充
    for i = 1:256
        if( srcHist(i,j) < dstHist(i,j))
            val = dstHist(i,j) - srcHist(i,j);
            tmpImg(excessPiexl(1:val)) = i-1;
            excessPiexl = excessPiexl(val+1:end);
            tmpImg(find(srcImg(:,:,j)==i-1)) = i-1;
        end
    end
```

```
    for i = 1:256
        if( srcHist(i,j) == dstHist(i,j))
            tmpImg(find(srcImg(:,:,j)==i-1)) = i-1;
        end
    end


    resultImg(:,:,j) = reshape(tmpImg,[srcH,srcW]);
end
```

实验结果

比直方图匹配，图片的左半部分“绿化”的更严重。