

作业 1: 直方图均衡化增强图片

王琛然 151220104 17721502736@163.com 17721502736

(南京大学 计算机科学与技术系, 南京 210093)

1 实现细节

因为图像的直方图呈均匀分布时, 对比度会有明显的增强, 所以直方图均衡化是指通过灰度变换函数, 将原图像直方图的分布均衡化, 在该实验中实现了均衡化函数, 对灰度图像和彩色图像都进行了图片增强, 而对彩色图的处理中, 用了三种方法, 下面会详细介绍。

1.1 灰度图像直方图均衡化

基本思路:

(1) 首先已知输入图片 A 的直方图分布, 并已知期望得到的图片 B 的直方图分布, 即为平均分布, 计算得到一个灰度变换函数(不写为代码), 推理过程如下:

核心思想: 利用面积相等

$$\int_0^{D_B} H_B(D)dD = \int_0^{D_A} H_A(D)dD$$

由于 D_B 是均匀分布的, 且大小为 A_0/L , 其中 A_0 是总的像素个数, L 是灰度级, 所以可以得到:

$$\int_0^{D_B} H_B(D)dD = \frac{D_B}{L} * A_0 = \frac{f(D_A)}{L} * A_0$$

即可得到灰度变换函数为:

$$f(D_A) = \frac{L}{A_0} * \int_0^{D_A} H_A(D)dD$$

(2) 得到图片 A 的直方图分布:

利用概率计算: 首先计算出各个像素值出现的次数, 并计算其概率, 即得到 H_A/A_0

```
% 统计每个像素值出现次数
cnt = zeros(1, 256);
for i = 1 : R
    for j = 1 : C
        cnt(1, input_channel(i, j) + 1) = cnt(1, input_channel(i, j) + 1) + 1;
    end
end

f = zeros(1, 256);
f = double(f);
cnt = double(cnt);

% 统计每个像素值出现的概率
for i = 1 : 256
    f(1, i) = cnt(1, i) / (R * C);
end
```

(3) 代码实现灰度变换函数，并得到均衡化后的直方图并输出图片

```
% 计算积分
for i = 2 : 256
    f(1, i) = f(1, i - 1) + f(1, i);
end

% 实现像素值[min, max]的映射
for i = 1 : 256
    f(1, i) = f(1, i) * (max - min);
end

% 每个像素点进行映射
input_channel = double(input_channel);
for i = 1 : R
    for j = 1 : C
        input_channel(i, j) = f(1, input_channel(i, j) + 1);
    end
end
```

1.2 彩色图片直方图均衡化

1.2.1 RGB

一种对彩色图片直方图均衡化的方法是将 RGB 图像的 R、G、B 三个通道，同灰度图作为二维矩阵，分别进行直方图均衡化，然后再拼接成 RGB 图像

```
r=input_image(:,:,:1);
v=input_image(:,:,:2);
b=input_image(:,:,:3);
r1 = hist_equal(r);
v1 = hist_equal(v);
b1 = hist_equal(b);
[output1] = cat(3,r1,v1,b1); %RGB
```

1.2.2 HSV

第二种办法是将 RGB 图像转为 HSV 图像。所谓 HSV 图像即为用色彩(H)，深浅(S)，明暗(V)描述。只对 V 通道进行直方图均衡化，并与原 H 与 S 拼接，再转为 RGB 图像

(注意，在这里 RGB 图像转为 HSV 图像时，进行了归一化，所以 HSV 矩阵中存储的值都为小数，需要将其转为整数再调用上面的直方图均衡化函数)

```
function [output] = hsv(input_channel)
hv=rgb2hsv(input_channel);
H=hv(:,:,:,1);
S=hv(:,:,:,2);
V=hv(:,:,:,3);
[X, Y] = size(V);
for i = 1:X
    for j = 1:Y
        V(i, j) = V(i, j)*255;
    end
end
R = hist_equal(V);
R = mat2gray(R);
F = cat(3, H, S, R);
[output] = hsv2rgb(F);
```

1.2.3 CLAHE

CLAHE 是对对比度有限的自适应直方图均衡，而限制对比度，实则限制灰度变换函数的斜率，即灰度直方图 Hist 的积分，也就是 Hist 的幅度。所以该算法主要分为两个部分：分块与裁剪，但是由于分块进行裁剪，最终图像会呈块状效应，则需要插值运算来解决

(1) 分块

将原图像水平方向分为 8 份，垂直方向分为 4 份，在此，为了能够使原图的长宽被整除。需要建立一个稍大的图像，将长宽补全保证整除，建立灰值查找表将图片像素值归于 $[\min V, \max V]$ ，并以块为单位，构建直方图

```

NrX = 8;
NrY = 4;
HSize = ceil(h/NrY);
WSize = ceil(w/NrX);

deltay = NrY*HSize - h;
deltax = NrX*WSize - w;

tmpImg = zeros(h + deltay, w + deltax);
tmpImg(1:h, 1:w) = input_channel;

new_w = w + deltax;
new_h = h + deltay;

NrPixels = HSize * WSize;

NrBins = 256;

LUT = zeros(maxV+1,1);

for i=minV:maxV
    LUT(i+1) = fix(i - minV);
end

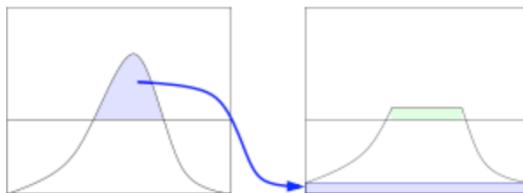
Bin = zeros(new_h, new_w);
for m = 1 : new_h
    for n = 1 : new_w
        Bin(m,n) = 1 + LUT(tmpImg(m,n) + 1);
    end
end

Hist = zeros(NrY, NrX, 256);
for i=1:NrY
    for j=1:NrX
        tmp = uint8(Bin(1+(i-1)*HSize:i*HSize, 1+(j-1)*WSize:j*WSize));
        for k = 1: 256
            Hist(i, j, k) = imhist(tmp, k);
        end
    end
end

```

(2) 裁剪

对子块中的直方图进行裁剪，高于某个上限幅值的面积均匀的分布在在整个灰度区间上



a. 计算直方图中高于裁剪值 ClipLimit 的部分 NrExcess

b. 以 $\text{upper} = \text{ClipLimit} - L$ 为界限对直方图进行处理：

- a) 若幅值高于 ClipLimit, 直接置为 ClipLimit
 - b) 若幅值处于 upper 和 ClipLimit 之间, 将其填补至 ClipLimit
 - c) 若幅值低于 upper, 直接填补 NrExcess/Nrbins 个像素点
- c. 处理剩余的像素点

代码:

```

function [Hist] = clipHistogram(Hist,NrBins,ClipLimit,NrX,NrY)
for i = 1: NrX
    for j = 1: NrY
        % 计算所有超出部分像素的和
        NrExcess = 0;
        for k = 1: NrBins
            excess = Hist(i,j,k) - ClipLimit;
            if excess > 0
                NrExcess = NrExcess + excess;
            end
        end

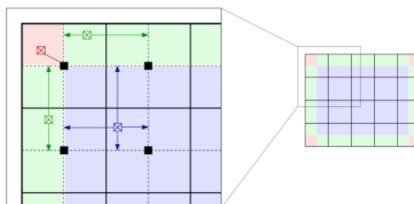
        % 裁剪分布图并将面积重新分配
        binIncr = NrExcess / NrBins;
        upper = ClipLimit - binIncr;
        for k = 1: NrBins
            % 如果超过限制, 直方图值为限制值
            if Hist(i,j,k) > Cliplimit
                Hist(i,j,k) = ClipLimit;
            else
                % 加上分配值会超过限制值
                if Hist(i,j,k) > upper
                    NrExcess = NrExcess + upper - Hist(i,j,k);
                    Hist(i,j,k) = ClipLimit;
                else
                    NrExcess = NrExcess - binIncr;
                    Hist(i,j,k) = Hist(i,j,k) + binIncr;
                end
            end
        end
    end

    if NrExcess > 0
        stepSize = max(1,fix(1 + NrExcess/NrBins));
        for k = 1: NrBins
            NrExcess = NrExcess - stepSize;
            Hist(i,j,k) = Hist(i,j,k) + stepSize;
            if NrExcess < 1
                break;
            end
        end
    end
end
end

```

(3) 线性插值

每个像素点处的值由它周围 4 个子块的映射函数值进行双线性插值得到



蓝色像素点处的值需要周围 4 个子块的映射函数分别做变换得到 4 个映射值, 再进行双线性插值; 对于红色像素点, 由于处于边角处, 只需一个子块的映射函数做变换; 对于绿色像素需要两个子块

(详细代码参加 code/clahe_function.m, 在此不再赘述)

2 结果

2.1 实验设置

使用 matlab2017 版，代码在 code 文件夹中，需要进行直方图均衡化的图片放在 asset 文件夹中，生成的结果放在 result 文件夹中，其中对于彩色图，result1 中存放分别对 RGB 三通道进行直方图均衡化的结果，result2 中存放将 RGB 转为 HSV 后，再对 V 通道进行直方图均衡化的结果，result3 存放 CLAHE 的结果，result4 存放 3 种方法与原图的比较结果；对于灰度图像则结果一样，只使用了 HE 均衡化方法

运行时只需运行 test_histeq.m 文件

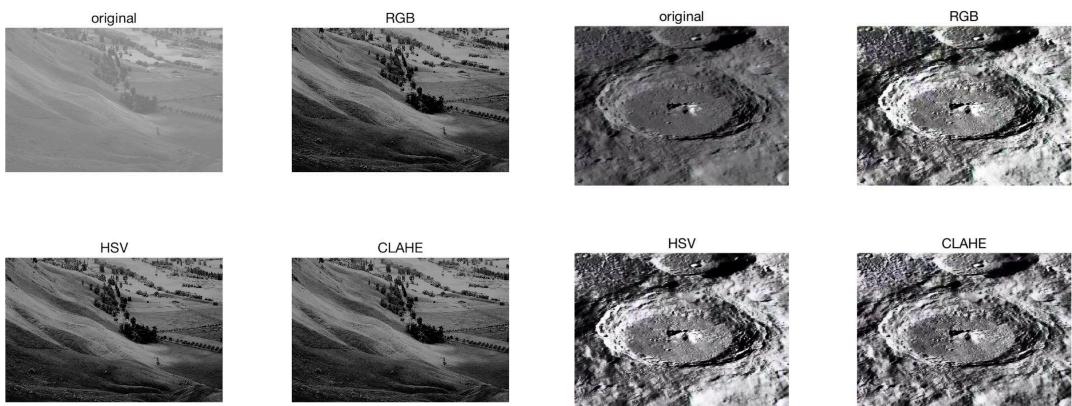
2.2 实验结果

(图片较多，在此仅展示几张，其他图片效果可以运行代码查看)

2.2.1 灰度图像：

gray.jpg 与 6.jpg：

(三张图只用了一种方法，所以只需关注与原图的区别)



可以看出，对灰度图像有明显的增强

2.2.2 彩色图像

1.jpg：



2.jpg:



color.jpg:



可以看出，使用分别对 RGB 图像各个通道进行直方图均衡化的方法在处理彩色丰富的图片是会出现色彩失真的现象；RGB 转 HSV 并对 V 通道直方图均衡化的方法对原图色彩的保持较好，但会使图片亮度较暗，但效果比第一种方法较好；CLAHE 方法在增强色彩的对比度的同时没有失真，明亮度也与原图基本一致，是三种方法中效果最好的一种。

3 参考

- [1] 对比度受限的自适应直方图均衡化(CLAHE) <https://blog.csdn.net/u010839382/article/details/49584181>
- [2] 基于直方图的图像增强算法 (HE、CLAHE) (1) <https://blog.csdn.net/baimafujinji/article/details/50614332>
- [3] 基于直方图的图像增强算法 (HE、CLAHE) (2) <https://blog.csdn.net/baimafujinji/article/details/50660189>
- [4] Contrast Limited Adaptive Histogram Equalization (CLAHE)
<https://cn.mathworks.com/matlabcentral/fileexchange/22182-contrast-limited-adaptive-histogram-equalization-clahe>