一、第一章 绪论

二、第二章 数字图像基础

三、第三章 灰度变换与空间滤波

3.1 直方图均衡化

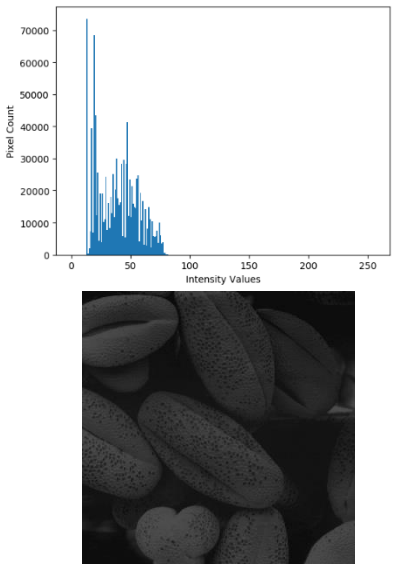
cv2.imread(path,flag)
height=width=1021

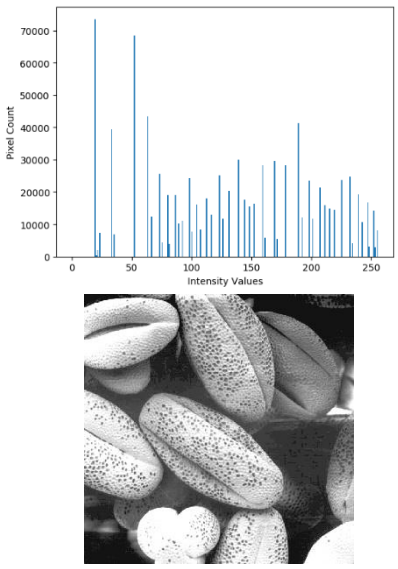
>>>hist, bin_edges = np.histogram([1, 2, 1], bins=[0, 1, 2, 3])
(array([0, 2, 1]), array([0, 1, 2, 3]))
input data--[1,2,1] range--bins=[0,1,2,3]

ceil 求最小整数
>>> a = np.array([-1.7, -1.5, -0.2, 0.2, 1.5, 1.7, 2.0])
>>> np.ceil(a)
array([-1., -1., -0., 1., 2., 2., 2.])

cumsum 求累计和
>>> a = np.array([[1,2,3], [4,5,6]])
>>> a
array([[1, 2, 3],
       [4, 5, 6]])
>>> np.cumsum(a)
array([1, 3, 6, 10, 15, 21])

解决 matplotlib 中中文字体显示问题输入：
import matplotlib
matplotlib.rcParams['font.family'] = 'SimHei'





3.2 直方图规定化

<table>
<tr><td colspan="3">dst=cv2.LUT(src,lut)</td></tr>
<tr><td colspan="3">LUT 函数用于查找表遍历图像，从数学上来看查找表是一个简单的一对一或多对一的函数，定义了如何将像素转换为新的值。从数据的组织关系上来看，查找表是一维或多维的数组，存储了不同输入值所对应的输出值。数据表在图像处理中主要用于像素的点运算，尤其是像素之间无位置相关性的操作中。</td></tr>
<tr><td>src</td><td colspan="2">原始图像的地址</td></tr>
<tr><td>lut</td><td colspan="2">查找表的地址，对于多通道图像的查找，它可以有一个通道，也可以与原始图像有相同的通道；</td></tr>
<tr><td>dst</td><td colspan="2">输出图像的地址</td></tr>
<tr><td>应用</td><td colspan="2">1.颜色空间缩减：将现有颜色空间值除以某个输入值，以获得较少的颜色数。例如，颜色值 0 到 9 可取为新值 0，10 到 19 可取为 10，以此类推。(示例来源于 OpenCV 官网)。
显然这是一个多对一的映射，I[new]＝I[old]/10*10。很容易想到，只要遍历图像矩阵的每一个像素，对像素应用上述公式就可以完成任务。只是这里用到了除法和乘法运算，而这两种运算又特别费时。鉴于一幅图像只涉及 256 个像素，我们大可开一个长度为 256 的数组，让其下标代表旧像素值，数组值代表新的像素值，如 lookup[256]={0,⋯,0,10,⋯,10,20,⋯,20,⋯,250,⋯,250}。这样我们遍历修改时不就可以通过像素值从表中查出要改变的像素值了么，而且这一过程只有赋值运算。
2.图像取反：反转图像的像素强度，使图像中的前景变为背景，背景变为前景。显然这是一个一对一的映射，即像素值 0 变为 255，1 变为 254⋯254 变为 1,255 变为 0。对应的查找表为 lookup[256]={255,254,⋯,1,0}。</td></tr>
<tr><td>代码示例</td><td colspan="2">

```
#include <iostream>
#include "opencv2/core/core.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"

using namespace std;
using namespace cv;

void Invert(Mat &img, const uchar* const lookup)
{
    int rows=img.rows;
    int cols=img.cols*img.channels();
    for(int i=0; i<rows; i++)
    {
        uchar *p=img.ptr<uchar>(i);
        for(int j=0; j<cols;j++)
            p[j]=lookup[p[j]];
    }
}

int main()
{
    Mat src=imread("test.jpg");  //将任意一张名为 test.jpg 的图片放置于工程文件夹 test 中
    if(!src.data)
    {
        cout<<"error! The image is not built!"<<endl;
        return -1;
    }
    // 为了演示效果，将图片转换成灰度图片
```

</td></tr>
</table>

```
        Mat img1=src;
        //cvtColor( src, img1, CV_RGB2GRAY );
        imshow("First",img1);
        //建立查找表
        uchar lookup[256];
        for(int i=0; i<256; i++)
            lookup[i]=255-i;
        //调用自定义图像取反函数
        Invert(img1, lookup);
        imshow("Second",img1);
        waitKey();
        return 0;
    }
```

```
#include<iostream>
#include<opencv2/core/core.hpp>
#include<opencv2/imgproc/imgproc.hpp>
#include<opencv2/highgui/highgui.hpp>

using namespace std;
using namespace cv;

int main()
{
    Mat src=imread("test.jpg");  //将任意一张名为 test.jpg 的图片放置于工程文件夹 test 中
    if(!src.data)
    {
        cout<<"error! The image is not built!"<<endl;
        return -1;
    }
    // 为了演示效果，将图片转换成灰度图片
    Mat img1=src;
    //cvtColor( src, img1, CV_RGB2GRAY );
    imshow("First",img1);
    //建立查找表
    Mat lookUpTable(1, 256, CV_8U);
    uchar *p = lookUpTable.data;
    for(int i=0; i<256; i++)
        p[i]=255-i;
    //通过 LUT 函数实现图像取反
    LUT(img1,lookUpTable,img1);

    imshow("Second",img1);
    waitKey();
    return 0;
}
```

参考	1. http://www.voidcn.com/article/p-dypyzpta-bgz.html 2.https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#cv2.LUT

利用 numpy 计算 PDF 及 CDF　PDF-概率密度函数（probability density function）
CDF-累计分布函数(Cumulative Distribution Function)

```
#利用 numpy 函数计算 PDF 及 CDF
def histCalculate_numpy(src):
    intensity_count = [0] * 256
    height,width = src.shape[:2]
    MN = height*width
    for i in range(0,height):
```

```
            for j in range(0,width):
                intensity_count[src[i][j]] += 1

        L = 256
        intensity_count,total_values_used = np.histogram(src.flatten(),L,[0,L])
        PDF_list = intensity_count*(L-1)/src.size
        CDF_list = PDF_list.cumsum()
        return CDF_list
```