基础

课程要求

- 一定是看完咱们关于知识点的讲解部分，再来听这节课。
- 有一个安装好了anaconda+vscode的环境。`pip install opencv-python`

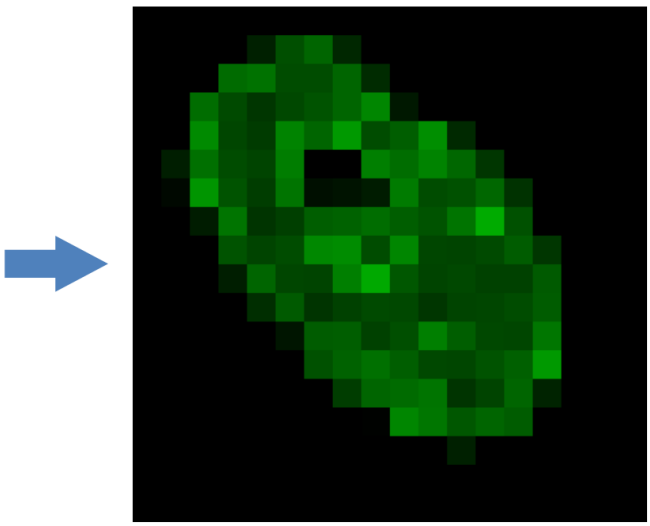
满分秘诀

- 一定不能用numpy和pandas等库
- opencv的库，只能用cv.read，和mat的一些赋值，转换的操作。**如果拿不准的，一定要问一下我，能不能用。**
- 核心是二维for循环，因为图像就是二维矩阵

前置知识

Displaying a digital image

0	2	2	2	5	8	11	8	2	2	0	0	0	0	0	0	0	0
0	0	2	11	76	136	164	85	11	5	2	2	0	0	0	0	0	0
0	2	25	172	181	133	133	164	90	14	5	2	2	0	0	0	0	0
2	5	175	130	104	127	141	164	206	65	31	11	2	2	0	0	0	0
2	28	212	124	110	204	164	232	133	155	218	87	14	2	2	0	0	0
2	73	178	133	121	195	34	31	198	175	204	167	104	14	5	0	0	0
2	45	226	141	113	184	53	59	70	192	133	138	167	99	11	2	0	0
0	2	70	184	102	116	155	161	175	155	141	184	255	138	34	5	2	0
0	0	5	141	121	133	209	215	133	206	124	121	130	153	104	8	2	0
0	0	2	73	164	124	121	198	252	147	121	127	119	119	150	19	2	0
0	0	0	5	93	150	102	119	130	127	104	121	124	133	153	25	2	0
0	0	0	0	5	62	153	155	119	136	198	155	127	124	187	19	2	2
0	0	0	0	0	5	138	161	178	155	127	124	141	158	232	5	2	2
0	0	0	0	0	0	11	113	164	172	184	102	121	164	79	2	2	2
0	0	0	0	0	0	2	5	36	206	187	147	164	153	5	2	2	0
0	0	0	0	0	0	0	0	2	5	25	76	31	2	2	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

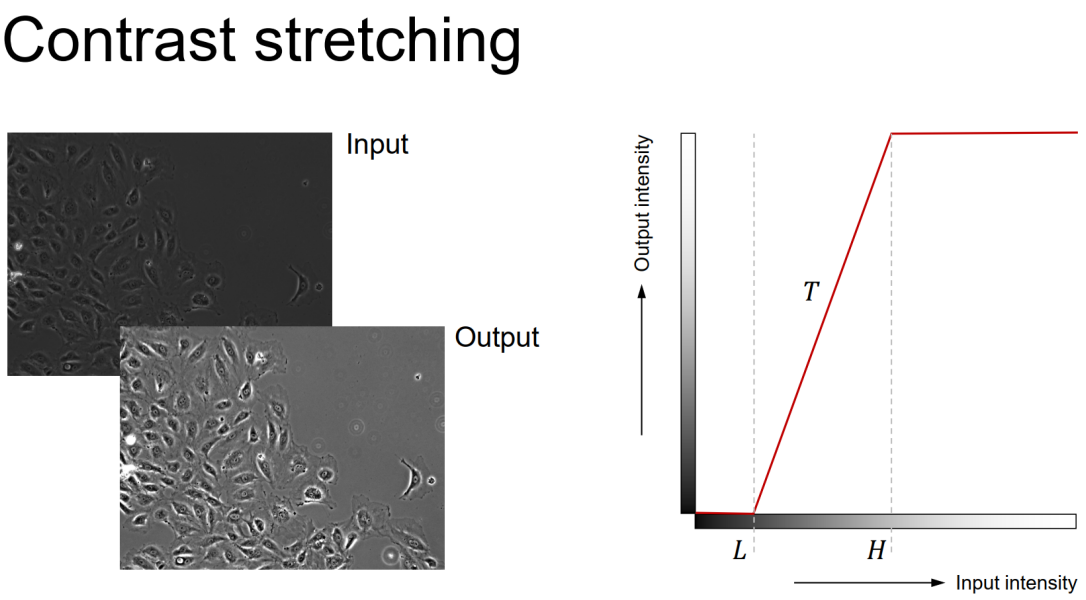


直方图

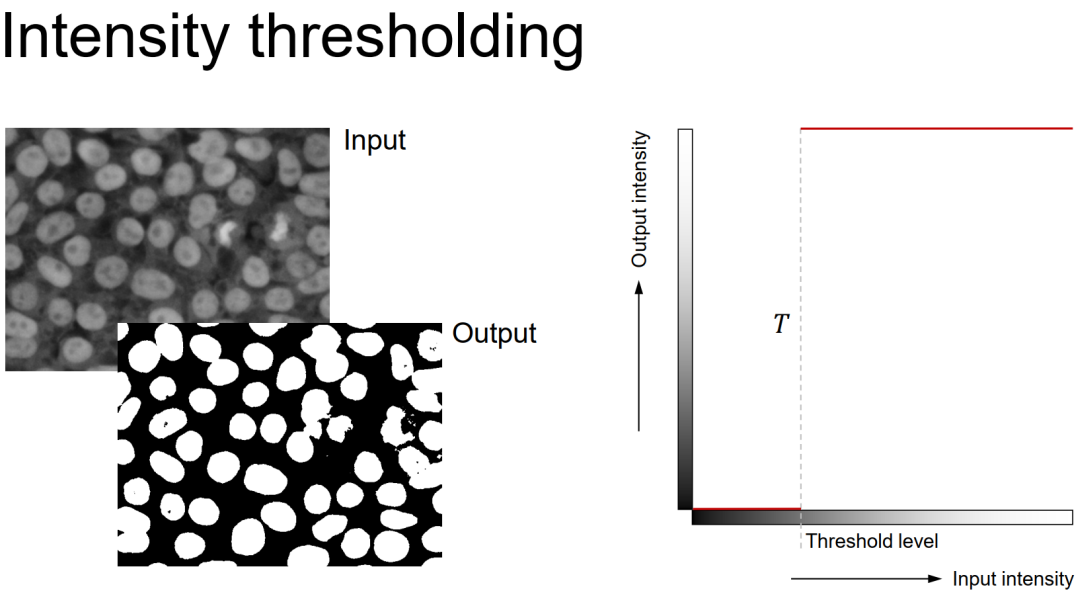
```
# 创建一个长度为256的数组，用于存储每个像素值的频数
    histogram = np.zeros(256, dtype=int)

    # 遍历图像的每个像素，并计算对应像素值的频数
    for row in range(image.shape[0]):
        for col in range(image.shape[1]):
            pixel_value = image[row, col]
            histogram[pixel_value] += 1
```

对比度拉伸



图像阈值化



咱们有库函数可以使用ostu,isodata等阈值化函数，咱们自己实现的函数，和官方的做对比，只要差不多，就是满分。

边缘检测

```
def compute_laplacian(image):
    # 定义Laplacian卷积核
    laplacian_kernel = np.array([[0, 1, 0],
                                 [1, -4, 1],
                                 [0, 1, 0]])
```

```
    # 获取图像尺寸
    height, width = image.shape

    # 创建一个与输入图像大小相同的输出图像
    laplacian_image = np.zeros((height, width), dtype=np.float32)

    # 对图像进行卷积操作
    for i in range(1, height - 1):
        for j in range(1, width - 1):
            # 应用卷积核计算每个像素的值
            pixel_value = (laplacian_kernel[0, 0] * image[i - 1, j
- 1] +
                           laplacian_kernel[0, 1] * image[i - 1, j]
+
                           laplacian_kernel[0, 2] * image[i - 1, j
+ 1] +
                           laplacian_kernel[1, 0] * image[i, j - 1]
+
                           laplacian_kernel[1, 1] * image[i, j] +
                           laplacian_kernel[1, 2] * image[i, j + 1]
+
                           laplacian_kernel[2, 0] * image[i + 1, j
- 1] +
                           laplacian_kernel[2, 1] * image[i + 1, j]
+
                           laplacian_kernel[2, 2] * image[i + 1, j
+ 1])

            # 将计算结果存储在输出图像中
            laplacian_image[i, j] = pixel_value
```