# 《计算机视觉》实验报告

## 姓名：刘彦辰 学号：21121319

## 实验 1

### 一. 任务 1

#### a) 核心代码：

```
# 定义名称添加函数，传入原始图像，返回添加完的图像
def setName(originImg):
    newImg = originImg.copy()
    cv2.putText(newImg, '21121319 Liu Yanchen', (4, 30), 4, 1, (255, 255, 255), 2, cv2.LINE_AA)
    return newImg

originImage = cv2.imread(r"img/tree.jpg", 1) # 读取图像
setName(originImage)  # 调用函数
```

#### b) 实验结果截图


#### c) 实验小结

根据文档调用库函数.

### 二. 任务 2

#### a) 核心代码：

```
[b, g, r] = cv2.split(originImg)
```

**b) 实验结果截图**

blue channel　　green channel　　red channel

**c) 实验小结**

利用 opencv 自带的库函数实现 BGR(蓝绿红)三通道分离, 产生灰度图. 灰度图中像素数值越大, 表示某种颜色越强.

## 三. 任务 3

**a) 核心逻辑:**

将蓝色通道的灰度图**(反)二值化(Binarization),** 在复制时避免复制背景(天空)的像素.

同时尝试使用 Sobel 算子处理图像后二值化(腐蚀 2 次,膨胀 5 次), 但效果较差, 弃用.

**b) 核心代码:**

图像处理:

```
    # 直接二值化
    [b, _, _] = cv2.split(originImage)  # 分离出蓝色通道
    (_, binarized_blue) = cv2.threshold(b, 240, 255, cv2.THRESH_BINARY)
    # 二值化
    binarized_blue = ~binarized_blue  # 取反

    # Sobel 算子计算梯度+二值化
    def sobel(img):
        # 灰度化
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        # 用 Sobel 算子计算梯度
        gradX = cv2.Sobel(gray, ddepth=cv2.CV_32F, dx=1, dy=0, ksize=-1)
        gradY = cv2.Sobel(gray, ddepth=cv2.CV_32F, dx=0, dy=1, ksize=-1)
        # subtract the y-gradient from the x-gradient
        gradient = cv2.subtract(gradX, gradY)
        gradient = cv2.convertScaleAbs(gradient)
        e01.showImg(gradient, "gradient")
        # 二值化
        (_, thresh) = cv2.threshold(gradient, 90, 255, cv2.THRESH_BINARY)
```

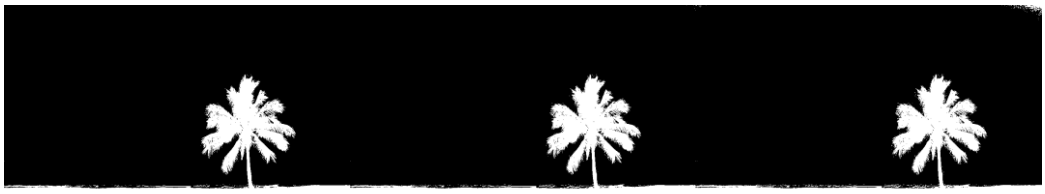
```
    # 形态学腐蚀与膨胀
    thresh = cv2.erode(thresh, None, iterations = 2)
    thresh = cv2.dilate(thresh, None, iterations = 5)
    return thresh
```

图像复制:

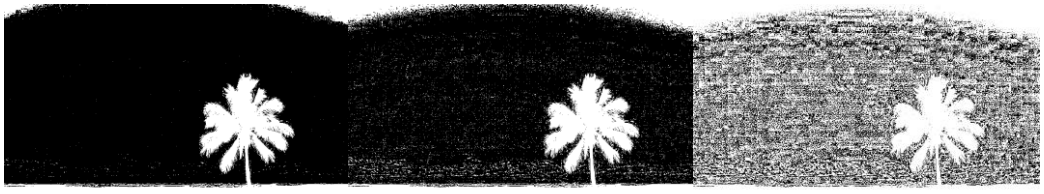
```
def copyTree_threshold(originImg, threshold, destY, destX, sourceY, sourceX, yLength, xLength, resizeY, resizeX):
    newImg = originImg.copy()
    tree = newImg[sourceY:sourceY + yLength, sourceX:sourceX + xLength]  # 截取树的矩阵
    treeThreshold = threshold[sourceY:sourceY + yLength, sourceX:sourceX + xLength]  # 截取树对应二值化矩阵
    resizedTree = cv2.resize(tree, (0, 0), None, resizeX, resizeY, interpolation=cv2.INTER_CUBIC)  # 缩放树矩阵
    resizedThreshold = cv2.resize(treeThreshold,(0,0),None,resizeX, resizeY, interpolation=cv2.INTER_CUBIC)  #缩放对应二值化矩阵
    # 遍历每个像素，当二值化像素点为 255 时(表示像素不属于天空)，将像素复制到相应位置
    for y in range(destY, destY + resizedTree.shape[0]):
        for x in range(destX, destX + resizedTree.shape[1]):
            if resizedThreshold[y-destY][x-destX] == 255:
                newImg[y][x] = resizedTree[y-destY][x-destX]
    return newImg
```

**c) 实验结果截图**

不同阈值下的蓝色通道反二值化截图:

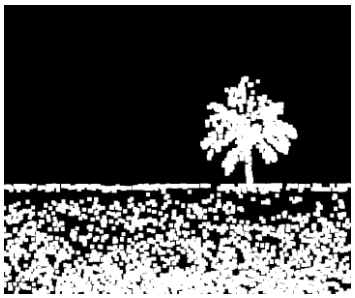


a) 220-255 b) 225-255 c) 230-255



d) 240-255 e) 245-255 f) 250-255

Sobel 算子二值化(效果较差，弃用):



g) Sobel

最终成品(选择了 230-255 的阈值):