计算机视觉实验报告

学号： 10185102144

姓名： 董辰尧

专业名称： 计算机科学与技术

学生年级： 2018 级本科

指导教师： 张倩

课程性质： 选修

研修时间： 2019～2020 学年第 2 学期

实验地点： 中山北路校区理科楼 B517

华东师范大学计算机科学与技术学院

2020 年 05 月 01 日

- 请自评你的项目完成情况，在表中相应位置划√。

实验工作自评

内容\评价	剖 析 功 能	参 考 完 成	较 大 修 改	独 特 功 能
1. Lab1		√		
2. Lab2		√		
3. Lab3		√		
4. Lab4		√		

.

总体自我评价

完成情况	尚 未 完 成	基 本 完成	较 好 完 成	圆 满 完 成
		√		

目　录

摘 要

在上半学期，我依据课堂授课内容以及网上的一些材料，在学习中进行了实践。让我能够更快速地掌握计算机视觉的相关专业知识。

因为 python 使用起来更加熟悉，Matlab 处理图像比较方便，所以，图像特征提取、二值图像的创建与特征计算、纹理特征计算方法、图像分割这 4 个实验我主要选择了 python 或者 Matlab 来完成。

实践 1：图像特征提取

一、目的

1) 了解实践 1 的任务
2) 了解实践 1 计划和安排
3) 掌握实践 1 软件的安装及环境配置方法
4) 掌握实践 1 的算法编写原理及调试方法

二、内容与设计思想

1) 下载软件并安装软件、配置环境
2) 了解图像特征提取的设计原理和基本思想
3) 练习图像特征提取程序设计的方法，实现图像特征提取功能

三、使用环境

Matlab online，Win10 操作系统环境

四、实验过程与分析、调试过程

特征提取是计算机视觉和图像处理中的一个概念。它指的是使用计算机提

取图像信息，决定每个图像的点是否属于一个图像特征。特征提取的结果是把 图像上的点分为不同的子集，这些子集往往属于孤立的点、连续的曲线或者连 续的区域。特征的好坏对泛化性能有至关重要的影响。

特征提取是图象处理中的一个初级运算，也就是说它是对一个图像进行的第一个运算处理。它检查每个像素来确定该像素是否代表一个特征。假如它是一个更大的算法的一部分，那么这个算法一般只检查图像的特征区域。作为特征提取的一个前提运算，输入图像一般通过高斯模糊核在尺度空间中被平滑。此后通过局部导数运算来计算图像的一个或多个特征。

1. 边缘检测

边缘是组成两个图像区域之间边界（或边缘）的像素。一般一个边缘的形状可以是任意的，还可能包括交叉点。在实践中边缘一般被定义为图像中拥有大的梯度的点组成的子集。一些常用的算法还会把梯度高的点联系起来来构成一个更完善的边缘的描写。这些算法也可能对边缘提出一些限制。

```
>> %边缘检测
I=imread('circuit.tif'); %读入图像
%进行边缘检测
BW1=edge(I,'prewitt'); %采用prewitt算子进行边缘检测
BW2=edge(I,'canny'); %采用canny算子进行边缘检测
subplot(1,3,1),imshow('circuit.tif')%显示
subplot(1,3,2),imshow(BW1);%显示
subplot(1,3,3),imshow(BW2)%显示
```

结果如下：

2. 角点检测

角是图像中点似的特征，在局部它有两维结构。早期的算法首先进行边缘检测，然后分析边缘的走向来寻找边缘突然转向（角）。后来发展的算法不再需要边缘检测这个步骤，而是可以直接在图像梯度中寻找高度曲率。后来发现这样有时可以在图像中本来没有角的地方发现具有同角一样的特征的区域。

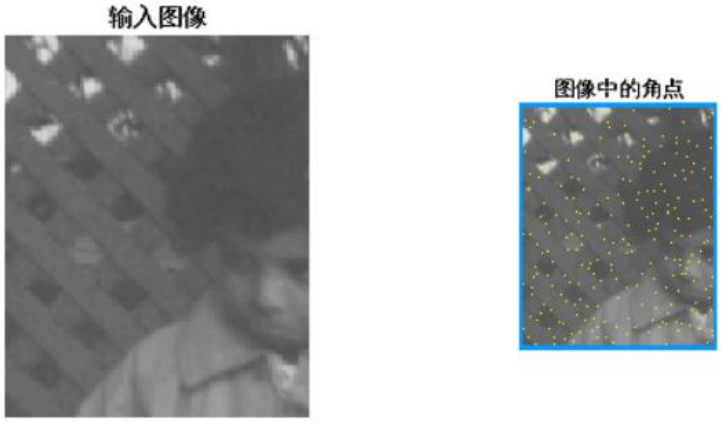
代码如下：

```
>> %Harris角点
%确定待检测的图像区域并显示
I=imread('pout.tif');
I=I(1:150,1:120);
subplot(1,2,1);
imshow(I);
title('输入图像');

%对图像进行Harris角点提取
CM=cornermetric(I);

%查找矩阵中最大值并显示
corner_peaks=imregionalmax(CM);
corner_idx=find(corner_peaks==true);
[r g b]=deal(I);
r(corner_idx)=255;
g(corner_idx)=255;
b(corner_idx)=0;
RGB=cat(3,r,g,b);
subplot(1,3,3);
imshow(RGB);
title('图像中的角点');
```

结果如下：



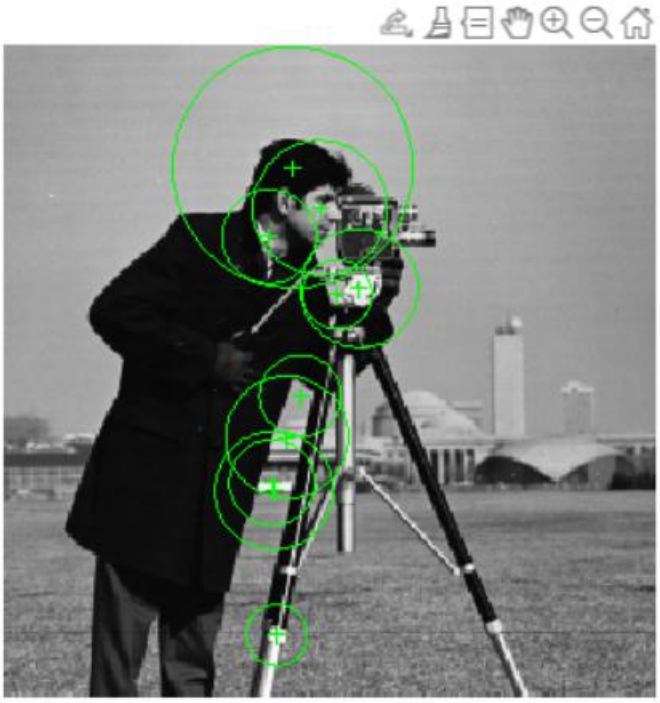
3. 区域检测

与角不同的是区域描写一个图像中的一个区域性的结构，但是区域也可能仅由一个像素组成，因此许多区域检测也可以用来监测角。一个区域监测器检测图像中一个对于角监测器来说太平滑的区域。区域检测可以被想象为把一张图像缩小，然后在缩小的图像上进行角检测。

代码如下：

```
>> %SURF特征提取
%读入图像;
I=imread('cameraman.tif');
%对输入的图像检测SURF特征;
points=detectSURFFeatures(I);
%显示最强的十个SURF特征点;
imshow(I);hold on;
plot(points.selectStrongest(10));
```

结果如下：



五、实验总结

在本次实验中，成功地实现了特征提取算法，提取了图像的各种特征，让我对一个图像有了新的认识，图像不再仅仅是图像，我会用观察其中的特征等角度去看待它

实践 2：二值图像的创建与特征计算

一、目的

1) 了解实践 2 的任务
2) 了解实践 2 计划和安排
3) 掌握实践 2 软件的安装及环境配置方法
4) 掌握实践 2 的算法编写原理及调试方法

二、内容与设计思想

1) 下载软件并安装软件、配置环境
2) 了解二值图像的创建与特征计算的设计原理和基本思想
3) 练习二值图像的创建与特征计算程序设计的方法，实现二值图像的创建与特征计算功能

三、使用环境

Matlab online，Win10 操作系统环境

四、实验过程与分析、调试过程

二值图像是指：每个像素点均为黑色或者白色的图像。二值图像一般用来描述字符图像，其优点是占用空间少，缺点是，当表示人物，风景的图像时，二值图像只能展示其边缘信息，图像内部的纹理特征表现不明显。这时候要使用纹理特征更为丰富的灰度图像。二值图像在计算机视觉 中有广泛的应用，例如车牌识别以及字符提取。

通过二值图像，可以提取很多特征，例如面积、欧拉数以及连通分量。

1. 二值图像生成

函数 im2bw：把图像转换成二值图像

```
>> load trees;
BW=im2bw(X,map,0.4);
subplot(121),imshow(X,map),title('原图像');
subplot(122),imshow(BW),title('二值图像');
```

结果如下：

2. 提取图像面积

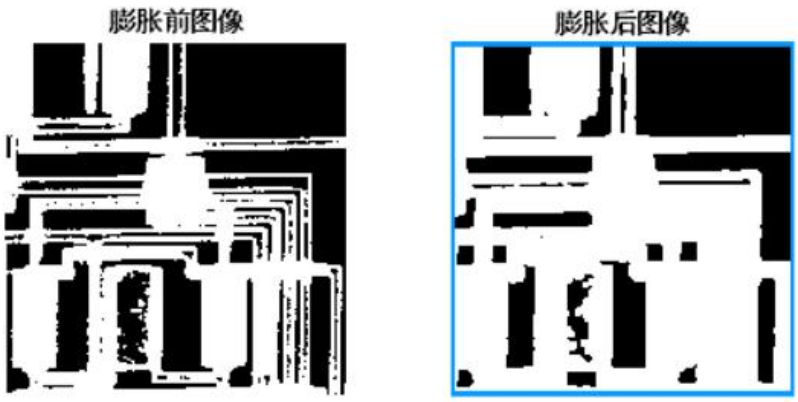
函数 bwarea:获取二值图像的面积

计算图像 circbw.tif 在膨胀运算前后图像面积的改变

主要代码如下：

```
>> BW=imread('circbw.tif');
SE=ones(5);
BW1=imdilate(BW,SE);
subplot(121),imshow(BW),title('膨胀前图像');
subplot(122),imshow(BW1),title('膨胀后图像');
increase=(bwarea(BW1)-bwarea(BW))/bwarea(BW)
```

结果如下：



```
increase =

    0.3456
```

3. 欧拉运算

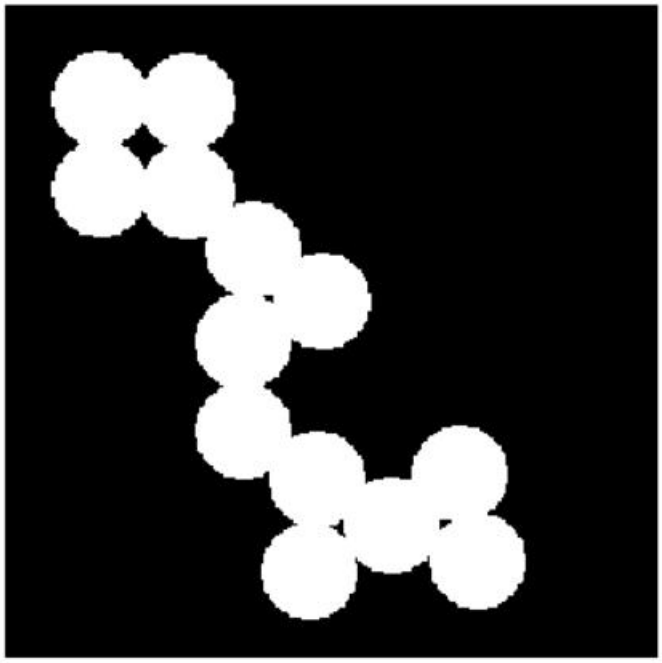
函数 bweuler:计算图像的欧拉数

计算 circles.png 的欧拉数

主要代码如下：

```
>> BW = imread('circles.png');
imshow(BW);
bwarea(BW)
```

结果如下：



```
ans =

   1.4187e+04
```

五、实验总结

在本次实验中，成功地实现了图像二值化以及特征计算算法，了解了图像二值化以及特征计算的设计原理和基本思想，增强了实践能力。

实践 3：纹理特征计算方法

一、目的

1) 了解实践 3 的任务
2) 了解实践 3 计划和安排
3) 掌握实践 3 软件的安装及环境配置方法
4) 掌握实践 3 的算法编写原理及调试方法

二、内容与设计思想

1) 下载软件并安装软件、配置环境
2) 了解纹理特征计算方法的设计原理和基本思想
3) 练习纹理特征计算方法程序设计的方法，实现纹理特征计算方法功能

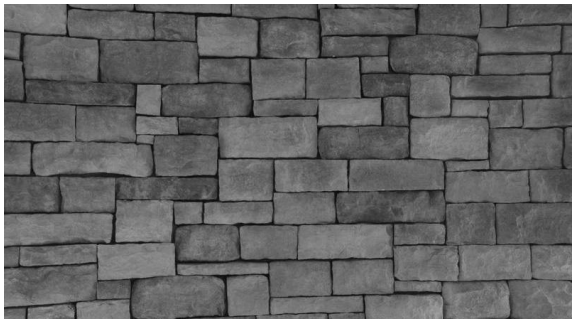
三、使用环境

Matlab online，Win10 操作系统环境

四、实验过程与分析、调试过程

图像纹理是指按照一定规则对元素或者基元进行排列所形成的重复模式，在数字图像处理以及计算机视觉领域有广泛应用。

我给出两张张有明显纹理特征的图片，如下图：



使用下面的代码对这两个图片进行计算：

```
>> close all;clear all;clc;
% 纹理图像的灰度差分统计特征
J = imread('qiang.jpg');
A = double(J);
[m,n] = size(A);
B = A;
C = zeros(m,n);
for i=1:m-1
    for j=1:n-1
        B(i,j) = A(i+1,j+1);
        C(i,j) = abs(round(A(i,j)-B(i,j)));
    end
end
h = imhist(mat2gray(C))/(m*n);
mean = 0;con=0;ent=0;  %均值mean，对比度con，熵ent
for i=1:256
    mean = mean + (i*h(i))/256;
    con = con+i*i*h(i);
    if(h(i)>0)
        ent = ent-h(i)*log2(h(i));
    end
end
mean,con,ent
```

得到结果如下：

qiang1.jpg　　qiang2.jpg

mean =　　mean =

0.0782　　0.0585

con =　　con =

1.0765e+03　　430.5984

ent =　　ent =

4.4663　　3.9879

可以看到 qiang1.jpg 的对比度更高；熵值更高，图像更加混乱；均值更大，图像看起来颜色偏深一点。

五、实验总结

在本次实验中，成功地实现了基于纹理的图像分割的算法，了解了图像纹理分析的设计原理和基本思想，增强了实践能力。

实践 4：图像分割

一、目的

1) 了解实践 4 的任务

2) 了解实践 4 计划和安排

3) 掌握实践 4 软件的安装及环境配置方法

4) 掌握实践 4 的算法编写原理及调试方法

二、内容与设计思想

1) 下载软件并安装软件、配置环境

2) 了解图像分割的设计原理和基本思想

3) 练习图像分割程序设计的方法，实现图像分割功能

三、使用环境

Jupyter notebook 软件-python3.7 版本，Win10 操作系统环境

四、实验过程与分析、调试过程

1. 分割

令 R 表示整个图像，P 代表某种相似性准则。一种区域分裂方法是首先将图像等分为 4 个区域，然后反复将分割得到的子图像再次分为 4 个区域，直到对任意 Ri，P(Ri)= TRUE，表示区域 Ri 已经满足相似性准则（比如该区域内灰度值相等或相似），此时不再进行分裂操作。这个过程可以用四叉树形式表示， 如下图所示：

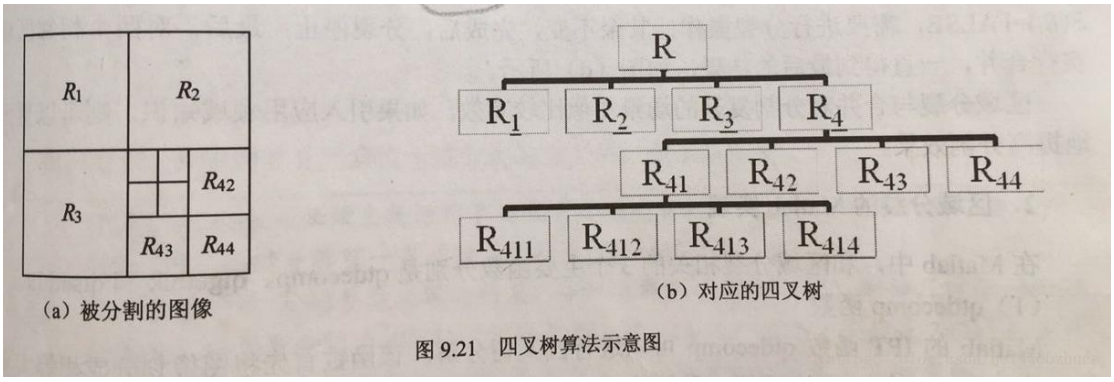


图 9.21 四叉树算法示意图

代码如下：

In [2]:

```python
#判断方框是否需要再次拆分为四个
def judge(w0, h0, w, h):
    a = img[h0: h0 + h, w0: w0 + w]
    ave = np.mean(a)
    std = np.std(a, ddof=1)
    count = 0
    total = 0
    for i in range(w0, w0 + w):
        for j in range(h0, h0 + h):
        #注意！我输入的图片数灰度图，所以直接用的img[j,i]，RGB图像的话每个img像素是一个三维向量，不能直接与avg进行比较大小。
            if abs(img[j, i] - ave) < 1 * std:
                count += 1
            total += 1
    if (count / total) < 0.95:#合适的点还是比较少，接着拆
        return True
    else:
        return False
```

2.合并

分裂操作完成之后，结果中一般会包含具有满足相似性的相邻区域，这就需要将满足相似性条件的相邻区域进行合并。可在分裂完成之后，也可以在分裂的同时，对具有相似特征的相邻区域进行合并。一种方法是将图像中任意具有相似特征的相邻区域 Rj 和 Rk 合并，即如果 P（RjURk）=TRUE，则合并 Rj 和 Rk。合并的两个区域可以大小不同，即不在同一层。当无法再进行聚合或拆分时操作停止。

整个过程如下图所示：

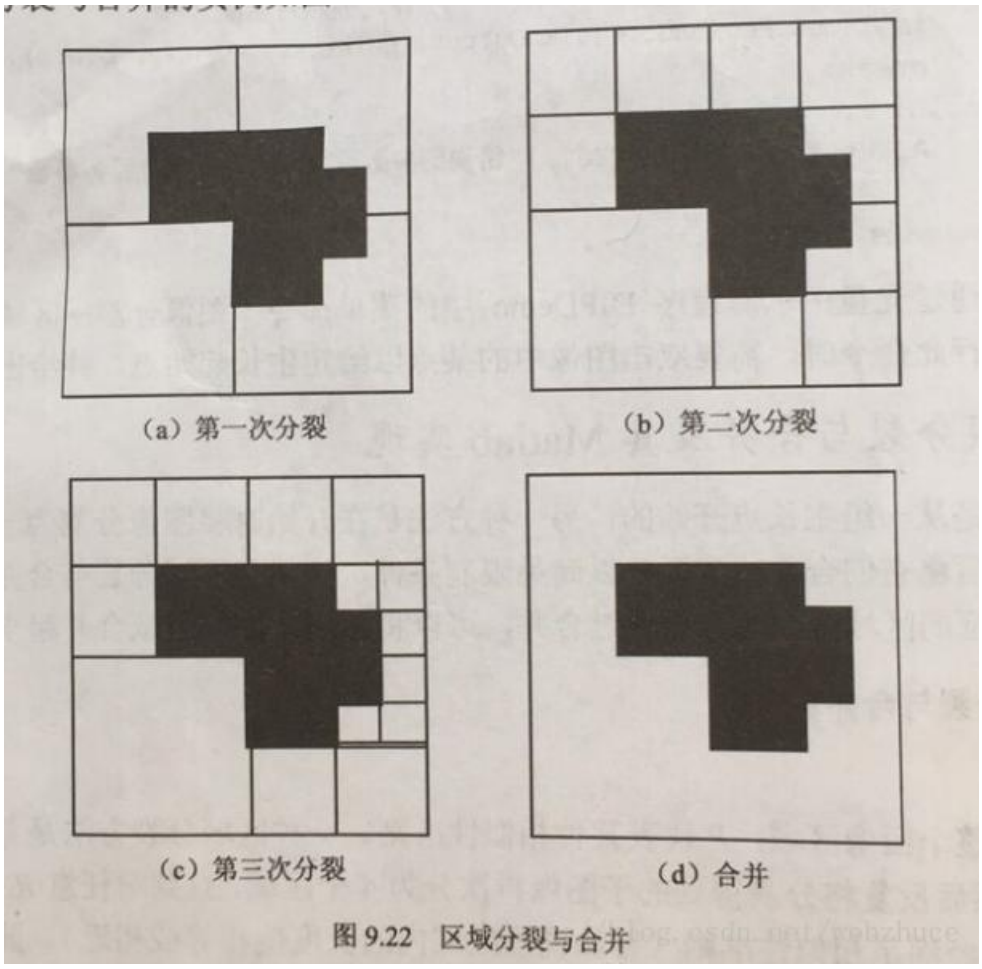
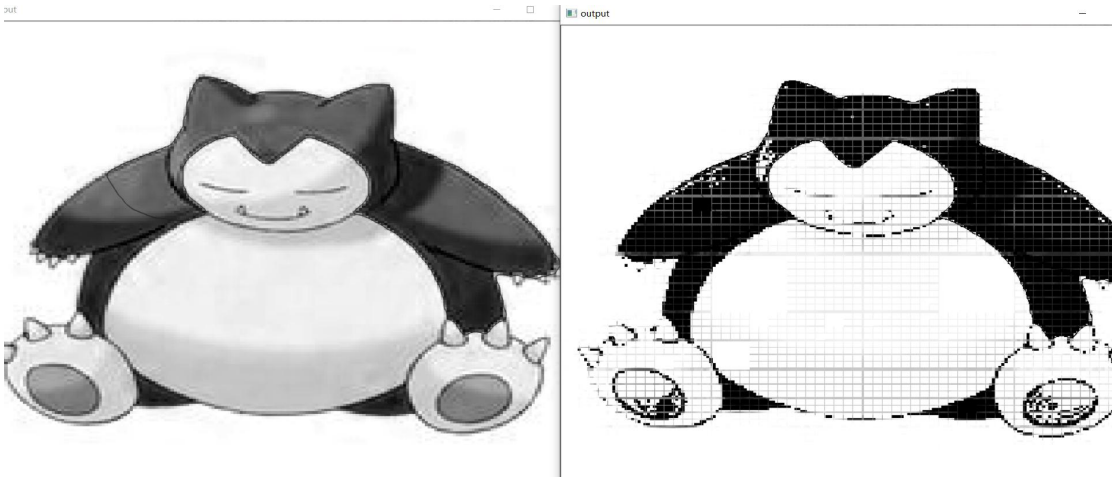


图 9.22　区域分裂与合并

代码如下：

```
def function(w0, h0, w, h):
    if judge(w0, h0, w, h) and (min(w, h) > 5):
        function(w0, h0, int(w / 2), int(h / 2))
        function(w0 + int(w / 2), h0, int(w / 2), int(h / 2))
        function(w0, h0 + int(h / 2), int(w / 2), int(h / 2))
        function(w0 + int(w / 2), h0 + int(h / 2), int(w / 2), int(h / 2))
    else:
        draw(w0, h0, w, h)
```

最后得到的结果如下：



五、实验总结

在本次实验中，成功地实现了基于水平集模型对图片分割的算法，了解了 水平集模型对图片分割算法的设计原理和基本思想。