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		√		
5. Lab5		√		
6. Lab6		√		
7. Lab7		√		

·

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：

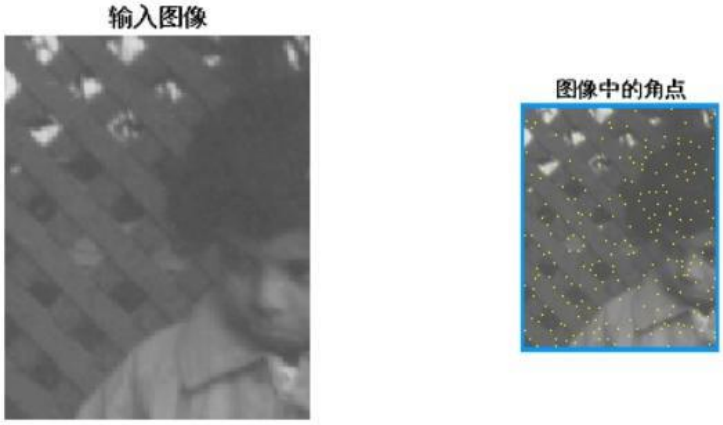
2. 角点检测

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

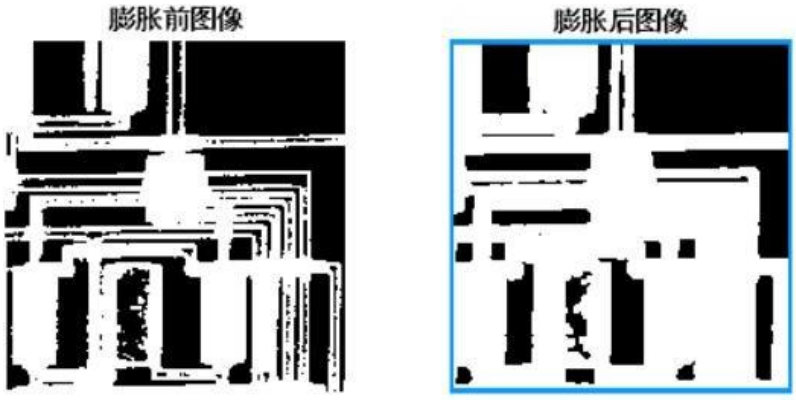
函数 bwarea:获取二值图像的面积

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

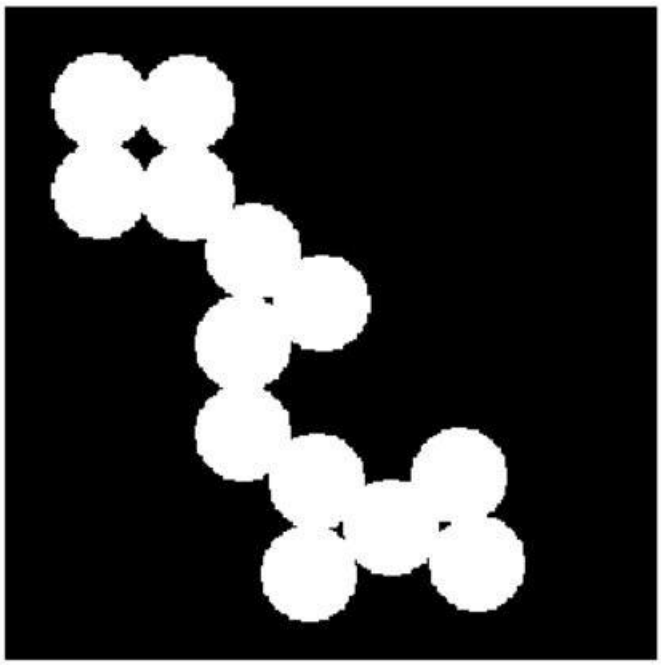
函数 bweuler:计算图像的欧拉数

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

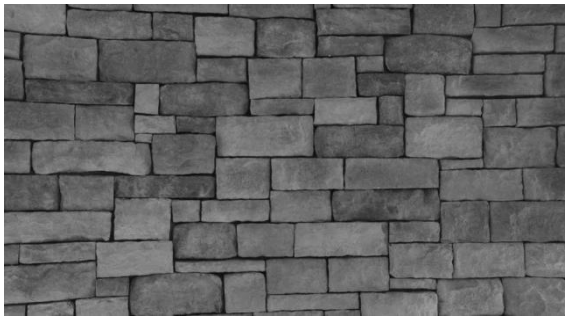
三、使用环境

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。

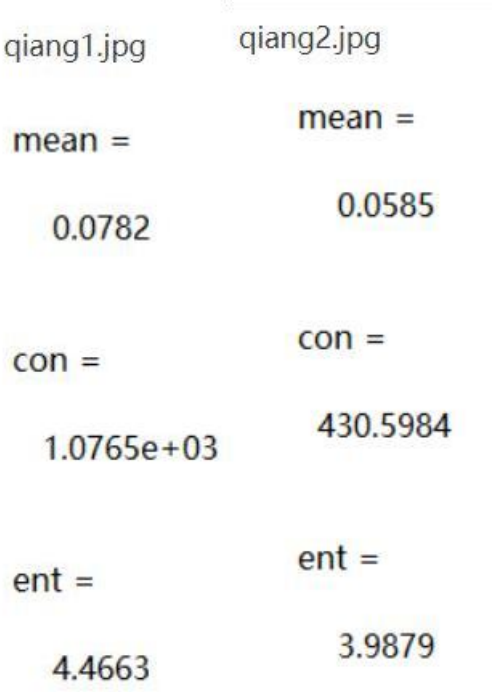
我给出两张张有明显纹理特征的图片，如下图：



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：



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：光流计算

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matlab软件并安装软件、配置环境

2）了解光流计算的设计原理和基本思想

3）练习光流计算程序设计的方法，实现光流计算功能

三、使用环境

matlab软件-win10操作系统环境

四、实验过程与分析、调试过程

光流法检测运动物体的原理：首先给图像中每个像素点赋予一个速度矢量（光流），这样就形成了光流场。如果图像中没有运动物体，光流场连续均匀，如果有运动物体，运动物体的光流和图像的光流不同，光流场不再连续均匀。从而可以检测出运动物体及位置。

应用背景：

根据图像前景和背景的运动，检测视频的变化，空间运动物体在观察成像平面上的像素运动的瞬时速度，是利用图像序列中像素在时间域上的变化以及相邻帧之间的相关性来找到上一帧跟当前帧之间存在的对应关系，从而计算出相邻帧之间物体的运动信息的一种方法。可以用来检测运动抖动物体

关键技术：

当人的眼睛观察运动物体时，物体的景象在人眼的视网膜上形成一系列连续变化的图像，这一系列连续变化的信息不断“流过”视网膜（即图像平面），好像一种光的“流”，故称之为光流（optical flow）。

本次实验我使用的是Horn-Schunck方法：

1. 首先是连续求解的过程：

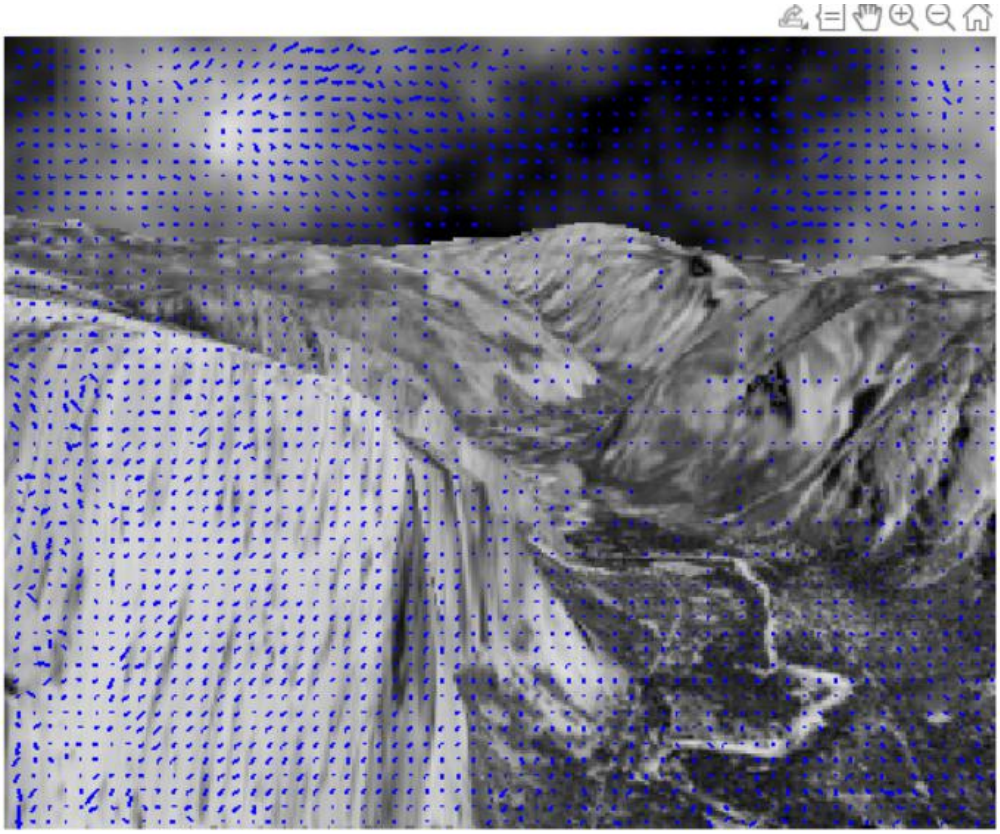
```
%% Default parameters
if nargin<1 || nargin<2
    im1=imread('yos9.tif');
    im2=imread('yos10.tif');
end
if nargin<3
    alpha=1;
end
if nargin<4
    ite=100;
end
if nargin<5 || nargin<6
    uInitial = zeros(size(im1(:,:,1)));
    vInitial = zeros(size(im2(:,:,1)));
elseif size(uInitial,1) ==0 || size(vInitial,1)==0
    uInitial = zeros(size(im1(:,:,1)));
    vInitial = zeros(size(im2(:,:,1)));
end
if nargin<7
    displayFlow=1;
end
if nargin<8
    displayImg=im1;
end

%% Convert images to grayscale
if size(size(im1),2)==3
    im1=rgb2gray(im1);
```

2. 接着是离散化迭代求解：

```
% Set initial value for the flow vectors
u = uInitial;
v = vInitial;

% Estimate spatiotemporal derivatives
[fx, fy, ft] = computeDerivatives(im1, im2);

% Averaging kernel
kernel_1=[1/12 1/6 1/12;1/6 0 1/6;1/12 1/6 1/12];

% Iterations
for i=1:ite
    % Compute local averages of the flow vectors
    uAvg=conv2(u,kernel_1,'same');
    vAvg=conv2(v,kernel_1,'same');
    % Compute flow vectors constrained by its local average and the optical flow constraints
    u= uAvg - ( fx .* ( ( fx .* uAvg ) + ( fy .* vAvg ) + ft ) ) ./ ( alpha^2 + fx.^2 + fy.^2);
    v= vAvg - ( fy .* ( ( fx .* uAvg ) + ( fy .* vAvg ) + ft ) ) ./ ( alpha^2 + fx.^2 + fy.^2);
end

u(isnan(u))=0;
v(isnan(v))=0;
```

五、实验总结

实验结果如下：



六、附录

Horn-Schunck Optical Flow Method - File Exchange - MATLAB Central

(mathworks.cn)

实践5：运动目标跟踪

一、目的

1）了解实践5的任务

2）了解实践5计划和安排

3）掌握实践5软件的安装及环境配置方法

4）掌握实践5的算法编写原理及调试方法

二、内容与设计思想

1）下载matlab软件并安装软件、配置环境

2）了解运动目标跟踪的设计原理和基本思想

3）练习运动目标跟踪程序设计的方法，实现运动目标跟踪功能

三、使用环境

win8.1+matlabR2015a

四、实验过程与分析、调试过程

代码来源：

http://www.cvl.isy.liu.se/en/research/objrec/visualtracking/scalvistrack/index.html

论文：Accurate Scale Estimation for Robust Visual Tracking(DSST)

将code下载至电脑任意位置，查看一下文件，进入目录D:\DSST_code\code，在D:\DSST_code\code\sequences\dog1内有自带的数据集可以不用下载数据了。

打开run_tracker.m，修改路径base_path = 'D:\DSST_code\code\sequences\';

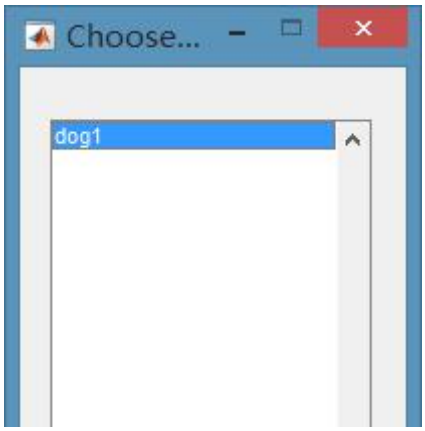
下面需要mex一下目录里的几个c++文件，打开compilemex_win.m里面有提供的mex命令模板，修改如下（需要适配自己电脑的opencv）：

mex -lopencv_core249 -lopencv_imgproc249 -L"E:\opencv\build\x64\vc10\lib" -I"E:\opencv\build\include" mexResize.cpp MxArray.cpp

将mexResize.cpp，MxArray.cpp放在你的matlab工作目录下，否则会提示找不到，中间两个路径需要改成电脑中opencv的位置，前两个文件需要根据版本修改数字值。

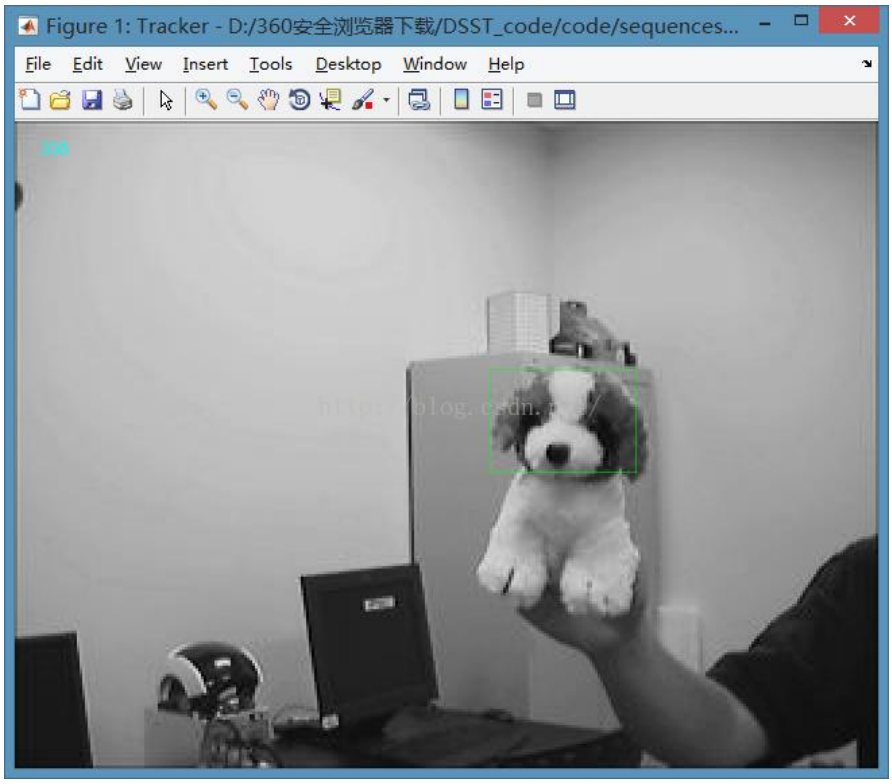
运行结果

运行成功会先出现如下GUI：



只有一个数据集可选，你也可以在目录里添加其他数据，但是要注意格式。

点击“OK”出现如下：



五、实验总结

这次实验比之前几次难度有所提升。首先代码比较难找，其次环境代码等配置复杂，在运行过程中出现了很多错误。这次实验之后，我不仅仅对运动检测有了新的认识，还对matlab更加熟悉了。

实践6：深度学习实践环境安装与配置

一、目的

1）了解实践6的任务

2）了解实践6计划和安排

3）掌握实践6软件的安装及环境配置方法

4）掌握实践6的算法编写原理及调试方法

二、内容与设计思想

1）下载anaconda软件并安装软件、配置环境

2）了解深度学习的设计原理和基本思想

3）练习深度学习程序设计的方法，实现图像分类功能

三、使用环境

Anaconda软件-3.6版本，Win10操作系统环境

四、实验过程与分析、调试过程

1.安装环境，在网上学习了torch的安装方法，安装结果如下

```
torch                                 1.8.1+cu102
torchaudio                            0.8.1
torchvision                           0.9.1+cu102
```

2.　　实现图像分类功能

我这里选择了手写图像识别的算法，先是把图像转化成文本

In [1]:

```
from PIL import Image, ImageDraw
import os
```

In [2]:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
from PIL import Image
# 将256灰度映射到16个字符上
def image_to_text(pixels, width, height):
    #symbols = "MNHQ$OC?7>!:-;. " # 16个字符
    symbols = list("01") # 这
    #symbols = "01" # 只映射到2个字符，改为01结果如何?
    string = ""
    for h in range(height):
        for w in range(width):
            rgb = pixels[w, h]
            string += symbols[int(sum(rgb) / 3.0 / 256.0 * len(symbols))]
        string += "\n"
    return string
    # 加载并调整大小
def load_and_resize_image(imgname, width, height):
    img = Image.open(imgname)
    if img.mode != 'RGB':
        img = img.convert('RGB')
    w, h = img.size
    rw = width * 1.0 / w
    rh = height * 1.0 / h
    r = rw if rw < rh else rh
    rw = int(r * w)
    rh = int(r * h)
    img = img.resize((rw, rh), Image.ANTIALIAS)
    return img
# 图片转为文本
def image_file_to_text(img_file_path, dst_width, dst_height):
    img = load_and_resize_image(img_file_path, dst_width, dst_height)
```

这里随便找了一个图，尝试转化

[3]:

```
imgfile = 'image/s.jpg'
w,h = 32,32
print(image_file_to_text(imgfile, w, h))
```

```
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111000001111111111111
11111111111000000000001111111111
11111111100000000000000111111111
11111111000001111110000011111111
11111110000011111111000001111111
11111110001111111111000011111111
11111110001111111111110111111111
11111110001111111111111111111111
11111110000111111111111111111111
11111110000001111111111111111111
11111110000000011111111111111111
11111110000000000111111111111111
11111111111000000000001111111111
11111111111111000000000111111111
11111111111111111000000011111111
11111111111111111110000001111111
11111111111111111111000001111111
11111111111111111111100001111111
11111111111111111111100001111111
11111110000111111111111000111111
11111110000111111111110000111111
11111110000011111111110000111111
11111110000001111111100000111111
11111111000000010000000011111111
11111111110000000000000111111111
11111111111100000000001111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
```

然后就是使用深度学习算法KNN，进行分类

数据预处理：

```
# 获取数据文件
fileList = os.listdir('./data/trainingDigits/')

# 定义数据标签列表
trainingIndex = []
# 添加数据标签
for filename in fileList:
    trainingIndex.append(int(filename.split('_')[0]))

# 定义矩阵数据格式
trainingData = np.zeros((len(trainingIndex),1024))
trainingData.shape#(3868, 1024)

# 获取矩阵数据
index = 0
for filename in fileList:
    with open('./data/trainingDigits/%s'%filename, 'rb') as f:

        # 定义一个空矩阵
        vect = np.zeros((1,1024))

        # 循环32行
        for i in range(32):
            # 读取每一行数据
            line = f.readline()

            # 遍历每行数据索引 line[j] 即为数据
            for j in range(32):
                vect[0,32*i+j] = int(line[j])

        trainingData[index,:] = vect
        index+=1
```

```
⌈[0. 0. 0. ... 0. 0. 0.⌉
```

KNN：

In [4]:

```
from sklearn.neighbors import KNeighborsClassifier

# 定义k为3个，即寻找最近的3个邻居
knn = KNeighborsClassifier(n_neighbors=3)

# 训练数据
knn.fit(trainingData,trainingIndex)
```

Out[4]:

```
KNeighborsClassifier(n_neighbors=3)
```

In [6]:

```
%%time
# 预测数据

predict_data = knn.predict(testData)
print(predict_data)
# Wall time: 7.8 s
knn.score(testData,testIndex)#0.9862579281183932
# 识别正确率：98.626%
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 4 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 7 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 9 9 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 1 6 6 6 1 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 3 8 8 8 8 0 8 5 8 8 8 8 8 1 1 8 8 1 8 8 8 9
 8 8 8 8 8 8 2 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 9 9 9 9 9 9 9 9 9 9 9 3 9 9 7 9 9 9 9 9 9 9 9 9 9]
Wall time: 2.81 s
```

Out[6]:

```
0.956221198156682
```

五、实验总结

本次实验用了一个简单的深度学习算法实现了图像的分类。在网上不是很容易

找到神经网络进行训练的图像分类，并且数据集也比较难下。本次实验是深度学习的入门，希望将来能在这个方面学到更多知识。

实践7：智能图像分割

一、目的

5）了解实践6的任务

6）了解实践6计划和安排

7）掌握实践6软件的安装及环境配置方法

8）掌握实践6的算法编写原理及调试方法

二、内容与设计思想

4）下载Matlab软件并安装软件、配置环境

5）了解智能图像分割的设计原理和基本思想

6）练习深度学习程序设计的方法，实现智能图像分割功能

三、使用环境

Matlab软件，Win10操作系统环境

四、实验过程与分析、调试过程

本次实验使用了模糊C均值算法（FCM），参考了matlab网站上的代码。

模糊c-均值聚类算法 fuzzy c-means algorithm (FCMA)或称（FCM）。在众多模糊聚类算法中，模糊C-均值（ FCM） 算法应用最广泛且较成功，它通过优化目标函数得到每个样本点对所有类中心的隶属度，从而决定样本点的类属以达到自动对样本数据进行分类的目的。

```
%% MMGR-WT achieves superpixel segmentation using adaptive and multiscale morphological
% L_seg is lable image with line
% i is the maximal iterations
% diff is the difference between the previous result and current gradient
function [L_seg,i,diff]=w_MMGR_WT(f,se_start)
% gauP=5 is the default value
% se_start=1;
max_itr=50;
min_impro=0.0001;
%% step1 gaussian filtering
sigma=1.0;gausFilter=fspecial('gaussian',[5 5],sigma);g=imfilter(f,gausFilter,'replicate
%% step2 compute gradient image
gg=colorspace('Lab<-RGB',g);
a1=sgrad_edge(normalized(gg(:,:,1))).^2;b1=sgrad_edge(abs(normalized(gg(:,:,2)))).^2;c1=
ngrad_f1=sqrt(a1+b1+c1);
%% step3 MMGR
f_g=zeros(size(f,1),size(f,2));diff=zeros(max_itr,1);
for i=1:max_itr
    gx=w_recons_CO(ngrad_f1,strel('disk',i+se_start-1));
    f_g2=max(f_g,double(gx));
    f_g1=f_g;f_g=f_g2;
    diff(i)=mean2(abs(f_g1 - f_g2));
    if i > 1
        if diff(i) < min_impro, break; end
    end
end
%% step4  watershed
L_seg=watershed(f_g);
```

假设样本集合为X={x1 ， x2 ， …， xn }，将其分成c 个模糊组，并求每组的聚类中心cj （ j=1，2，…，C），使目标函数达到最小。

下面是主函数实现

```
clear all
close all
cluster=2;
f_ori=imread('12003.jpg');
%MMGR.
SE=3;
L1=w_MMGR_WT(f_ori,SE);
L2=imdilate(L1,strel('square',2));
[~,~,Num,centerLab]=Label_image(f_ori,L2);
%% fast FCM
Label=w_super_fcm(L2,centerLab,Num,cluster);
Lseg=Label_image(f_ori,Label);
figure,imshow(Lseg);
```

最后得到的结果如下：

这个算法不够稳定，在我试验多次之后发现，有的时候对于海星的图片，他识别不出来海星的形状。

FCM算法步骤

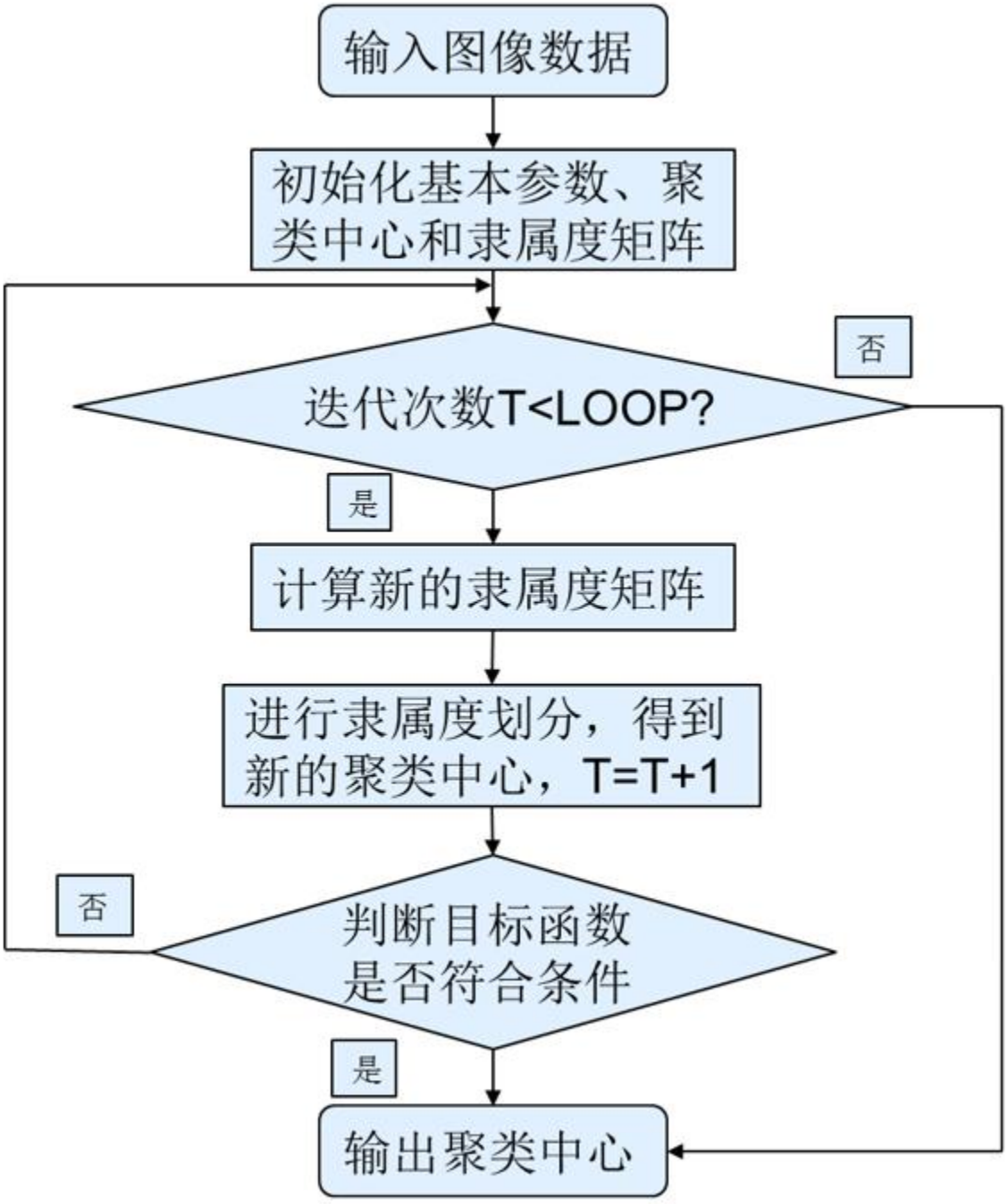
(1) 用值0,1之间的随机数初始化隶属度矩阵。

(2) 计算各聚类中心

(3) 计算价值函数，如果它小于某个确定的阈值，或它相对上次价值函数值得改变量小于某个阈值，则算法停止。

(4) 计算新的隶属度矩阵，返回步骤（2）。

上述算法也可以先初始化聚类中心，然后再执行迭代过程。由于不能确保FCM收敛于一个最优解。算法的性能依赖于初始聚类中心。因此，我们要么用另外的快速算法确定初始聚类中心，要么每次用不同的初始聚类中心启动该算法，多次运行FCM。



五、实验总结

本次实验用了FCM算法实现了图像分割。