Freeibox人工智能实验箱

实验教学课程

目 录

*说明：本实验手册仅收录部分实验课程，未收录的内容欢迎拨打 0731-89872400 咨询详情。

1　数字类型、转换、运算

一、实验目的

（1）掌握 4 种不同数字类型的定义；

（2）掌握不同数字类型之间的转换；

（3）掌握数字的基本运算；

（4）掌握字符与整型变量的相互转换。

二、实验内容

（1）编写程序，给定浮点数 num_float，显示该数据及类型；转换成整型，显示其数据及类型；

（2）编写程序，将两个数 2 和 3 组合成复数，并显示其数据及类型；

（3）编写程序，将字符串转换为整数，并显示其数据及类型；

（4）编写程序，将整数 65、90、97、122 转换为字符，将字符 A 转换为整数，并显示；

（5）编写程序，显示两个布尔量及类型；

（6）从键盘输入浮点型变量 x 的值，求方程 y=x^2+2x-10 所对应的 y 值并输出；

（7）编写程序，从键盘输入一个 4 位正整数，输出该数的反序数。例如 1234 的反序数是 4321。

三、算法分析

1. Python 支持三种不同的数字类型。

（1）整型(int)：通常被称为是整型或整数，是正或负整数，不带小数点。

（2）浮点型(float) ：浮点型由整数部分与小数部分组成，浮点型也可以使用科学计数法表示（2.5e2 = 2.5 x 10^2 = 250）

（3）复数(complex)：复数由实数部分和虚数部分构成，可以用 a + jb 或 complex(a,b) 表示，复数的实部 a 和虚部 b 都是浮点型。

（4）布尔型(bool)：是整型的子类型，提供了两个布尔值来表示真（对）或假（错），分别用 True（真或对）或 False（假或错）来表示。

2. 转换函数。

（1）int(x) 将 x 转换为一个整数。

（2）float(x) 将 x 转换到一个浮点数。

（3）eval(x)：将 x 转化为一个整数。

（4）complex(x, y):将 x 和 y 转换到一个复数，实数部分为 x，虚数部分为 y。x 和 y 是数字表达式。

（5）chr(x):将 x 转换为字符。

（6）ord(x)：将字符 x 转换为整数。

（7）eval(x)：将字符串转换为整数。

3. 运算符号。

**,~+-,*/%//@等。

四、实验步骤

1. 运行 jupyter lab

（1）在“实验”文件夹中空白处单击鼠标右键，选择“打开终端”，在终端界面输入“jupyter lab”。

（2）在 jupyter lab 编程界面选择 Notebook 菜单下的 Python3，进入程序编辑器。

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为所做实验名。

2.根据七个实验内容编写程序

⑴ 给定浮点数 num_float，显示数据及类型；转换成整型，显示转换后数据及类型。

参考代码:

```
num_float = 3.5
print(num_float)
print(type(num_float))
num_int = int(num_float)
print(num_int)
print(type(num_int))
```

运行结果:

```
3.5
<class 'float'>
3
<class 'int'>
```

(2) 将两个数 2 和 3 组合成复数，并显示其数据及类型。

参考代码:

```
num_complex = complex(2, 3)
print(num_complex)
print(type(num_complex))
```

运行结果:

```
(2+3j)
<class 'complex'>
```

(3) 将字符串转换为整数，并显示。

参考代码:

```
str_num = '789'
num = eval(str_num)
print(num)
print(type(num))
```

运行结果:

```
789
<class 'int'>
```

(4) 整数与字符相互转换，并显示；

参考代码:

```
print(chr(65))
print(chr(90))
print(chr(97))
print(chr(122))
print(ord('A'))
```

运行结果:

```
A
Z
a
z
65
```

(5) 显示布尔量及类型。

参考代码:

```
print(True) # True 首字母要大写
print(False) # False 首字母要大写
 print(type(True)) # 查看 True 的类型
print(type(False)) # 查看 False 的类型
```

运行结果:

```
True
False
<class 'bool'>
<class 'bool'>
```

(6) 从键盘输入浮点型变量 x，求方程 y=x^2+2x-10 所对应的 y 值并输出。

参考代码:

```
x=float(input('请输入浮点数：'))
y=x**2+2*x-10
print(y)
```

运行结果:

（1）在“实验”文件夹中空白处单击鼠标右键，选择“打开终端”，在终端界面输入“jupyter lab”。

（2）在 jupyter lab 编程界面选择 Notebook 菜单下的 Python3，进入程序编辑器。

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为所做实验名。

2.根据七个实验内容编写程序

(1)输入任意大的自然数 num，输出各位数字之和。

参考代码:

```
num = input('请输入一个自然数：')
print(sum(map(int, num)))
```

运行结果:

```
请输入一个自然数：1234
10
```

(2)输入两个集合 setA、setB，分别输出它们的交集、并集、差集。

参考代码:

```
setA = eval(input('请输入一个集合：'))
setB = eval(input('再输入一个集合：'))
print('交集：', setA & setB)
print('并集：', setA | setB)
print('setA-setB：', setA - setB)
```

运行结果:

```
请输入一个集合：{1,2,3,4}
再输入一个集合：{2,4,5,6}
交集： {2, 4}
并集： {1, 2, 3, 4, 5, 6}
setA-setB： {1, 3}
```

(3)输入一个自然数，输出它的二进制、八进制、十六进制表示形式；

参考代码:

```
num = int(input('请输入一个自然数：'))
print('二进制：', bin(num))
print('八进制：', oct(num))
print('十六进制：', hex(num))
```

运行结果:

2 Python 运算符、内置函数、序列基本用法

一、实验目的

（1）熟练运用 Python 运算符；

（2）熟练运用 Python 内置函数；

（3）了解 lambda 表达式作为函数参数的用法；

（4）了解列表、元组、字典、集合的概念和基本用法；

二、实验内容

（1）编写程序，输入任意大的自然数，输出各位数字之和；

（2）编写程序，输入两个集合 setA、setB，分别输出它们的交集、并集、差集；

（3）编写程序，输入一个自然数，输出它的二进制、八进制、十六进制表示形式；

（4）编写程序，输入一个包含若干整数的列表，输出一个新列表，要求新列表只包含原列表中的偶数；

（5）编写程序，输入两个分别包含若干整数的列表 lstA 和 lstB，输出一个字典，要求使用列表 lstA 中的元素作为键，列表 lstB 中的元素作为值，并且最终字典中的元素数量取决于 lstA 和 lstB 中元素最少的列表的数量；

（6）编写程序，输入一个包含若干整数的列表，输出新列表，要求新列表中的所有元素来自于输入的列表，并且降序排列。

（7）编写程序，输入一个包含若干整数的列表，输出列表中所有整数连乘的结果。

三、算法分析

（1） sum(map(int,num)):将 num 中每个元素转换成整数，再求和；

（2）集合的交集运算符（&）、并集运算符（|），差集运算符（-）；

（3）十进制转二进制、八进制、十六进制函数分别为 bin()、oct()、hex();

（4）list(filter(lambda x: x%2==0, lst))：筛选列表 lst 中的偶数；

（5）dict(zip(lstA, lstB))：将两个列表 lstA、lstB 组合成字典，长度等于短序列的长度；

（6） sorted(lst, reverse=True)：将列表 lst 降序排列；

（7）reduce(lambda x,y: x*y, lst) 将列表 lst 中的数连乘，先取 lst 中前两个数相乘，得到的结果再与第 3 个数相乘，依次类推，最后实现连乘;

四、实验步骤

1.运行 jupyter lab

```
请输入一个自然数：125
二进制：  0b1111101
八进制：  0o175
十六进制：  0x7d
```

(4) 输入一个包含若干整数的列表 lst，筛选列表 lst 中的偶数，输出一个只包含偶数的新列表；

参考代码:

```
lst = input('请输入一个包含若干整数的列表：')
lst = eval(lst)
print(list(filter(lambda x: x%2==0, lst)))
```

运行结果:

```
请输入一个包含若干整数的列表：[1,2,3,4,5,6,7,8,]
[2, 4, 6, 8]
```

(5) 输入两个包含若干整数的列表 lstA 和 lstB，将它们组合成字典输出，长度等于短序列的长度。

参考代码:

```
lstA = eval(input('请输入包含若干整数的列表 lstA：'))
lstB = eval(input('请输入包含若干整数的列表 lstB：'))
result = dict(zip(lstA, lstB))
print(result)
```

运行结果:

```
请输入包含若干整数的列表lstA：['name','age','sex']
请输入包含若干整数的列表lstB：['Tom',20,'male','big']
{'name': 'Tom', 'age': 20, 'sex': 'male'}
```

(6)输入一个包含若干整数的列表，降序排列并输出。

参考代码:

```
lst = eval(input('请输入包含若干整数的列表 lst：'))
print(sorted(lst, reverse=True))
```

运行结果:

```
请输入包含若干整数的列表lst：[1,3,2,6,5,7,4]
[7, 6, 5, 4, 3, 2, 1]
```

(7)导入 Python 标准库 functools 模块中的 reduce 函数，输入一个包含若干整数的列表，输出列表中所有整数连乘的结果。

参考代码:

```
from functools import reduce
lst = eval(input('请输入包含若干整数的列表 lst：'))
print(reduce(lambda x,y: x*y, lst))
```

运行结果:

```
请输入包含若干整数的列表lst：[1,2,3,4,5,6]
720
```

```
请输入浮点数：2.5
1.25
```

(7) 输入一个 4 位正整数，输出该数的反序数。

参考代码:

```
x=int(input('请输入一个四位正整数:'))
t1=x%10
t2=x%100//10
t3=x%1000//100
t4=x//1000
y=t4+t3*10+t2*100+t1*1000
print('{}的反序数是{}'.format(str(x),str(y)))
```

运行结果:

```
请输入一个四位正整数:2465
2465的反序数是5642
```

3　程序选择结构实验

——快速判断一个数是否为素数

一、实验目的

（1）掌握单分支结构的 if 语句；

（2）掌握双分支结构的 if…else 语句；

（3）掌握多分支结构的 if…else…else 语句；

（4）理解素数判断方法；

二、实验内容

编写程序，快速判断一个数是否为素数。素数又称为质数，定义为一个大于 1 的自然数 n，除了 1 和它自身外，不能被其他自然数整除的数。

三、算法分析

素数只能被 1 和它自身整除，合数 n 进行因数分解得到的两个因数一定是一个小于$\sqrt{n}$，另一个大于$\sqrt{n}$。

素数快速判断方法：2，3，5 是素数；偶数不是素数；大于 5 的自然数 n 除以 6，余数为 0、2、3、4 肯定不是素数，余数为 1 和 5 则需进一步判断 n 能否被{3，5，7…int($\sqrt{n}$)+1}当中的一个数整除，若能，则 n 是合数，不是素数。

程序采用多分支结构实现该算法。

四、实验步骤

1.运行 jupyter lab

（1）在“实验”文件夹中空白处单击鼠标右键，选择“打开终端”，在终端界面输入“jupyter lab”。

（2）在 jupyter lab 编程界面选择 Notebook 菜单下的 Python3，进入程序编辑器。

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为所做实验名。

2.输入一个大于 1 的自然数 n，若 n=2/3/5，则为素数；

参考代码:

```
n=input("Input an integer")
n = int(n)
if n in [2,3,5]:
    print('Yes')
```

3.判断 n 是否为偶数，偶数不是素数；

参考代码:

```
elif n%2==0:
    print('No')
```

4. 将 n 对 6 取余，余数为 0、2、3、4 的不是素数；

参考代码:

```
else:
    #大于 5 的素数必然出现在 6 的倍数两侧
    #因为 6x+2、6x+3、6x+4 肯定不是素数，假设 x 为大于 1 的自然数
    m=n%6
    if m!=1 and m!=5:
        print('No')
```

5. 否则将 n 对{3，5，7…int($\sqrt{n}$)+1 依次取余，若有一个余数为 0，则不是素数。

参考代码:

```
  else:
        for i in range(3,int(n**0.5)+1,2):
            if n%i==0:
                print('No')
                break
        else:
            print('Yes')
```

五、运行效果示例

```
Input an integer:88
No
```

输入：88，不是素数

```
Input an integer:23
Yes
```

输入：23，是素数

4 程序循环结构实验

——模拟决赛现场最终成绩计算过程

一、实验目的

（1）掌握 while 循环结构；

（2）掌握 for 循环结构；

（3）掌握循环控制语句；

（4）熟练使用循环嵌套；

二、实验内容

编写程序，模拟决赛现场最终成绩计算过程。首先输入大于 2 的整数作为评委人数，然后依次输入每个评委的打分，要求每个分数都介于 0~100。最终成绩为去掉一个最高分，去掉一个最低分后剩余分数的平均值。采用 while 循环、for 循环及循环嵌套实现。

三、算法分析

决赛现场最终成绩：去掉一个最高分，去掉一个最低分，剩余分数的平均值。

三、实验步骤

1.运行 jupyter lab

（1）在“实验”文件夹中空白处单击鼠标右键，选择“打开终端”，在终端界面输入“jupyter lab”。

（2）在 jupyter lab 编程界面选择 Notebook 菜单下的 Python3，进入程序编辑器

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为所做实验名。

2.输入评委人数，评委人数必须大于 2；

参考代码:

```
while True:
    try:
        n = int(input('请输入评委人数：'))
        assert n>2
        break
    except:
        print('必须输入大于 2 的整数')
```

```
```

3. 初始化 maxScore，minScore，total 变量，依次输入每个评委的打分;

参考代码:

```
maxScore, minScore = 0, 100
total = 0

# 依次输入每个评委的打分
for i in range(n):
    # 这个 while 循环用来保证用户必须输入 0 到 100 之间的数字
    while True:
        try:
            score = float(input('请输入第{0}个评委的分数：'.format(i+1)))
            assert 0<=score<=100
            break
        except:
            print('必须属于 0 到 100 之间的实数.')
```

4.累加评分，找出最高分和最低分;

参考代码:

```
total += score
    if score > maxScore:
        maxScore = score
    if score < minScore:
        minScore = score
```

5.去掉最高分和最低分，计算剩余分数的平均值，得最终成绩，并显示。

参考代码:

```
# 计算平均分，保留 2 位小数
finalScore = round((total-maxScore-minScore)/(n-2), 2)

formatter = '去掉一个最高分{0}\n 去掉一个最低分{1}\n 最后得分{2}'
print(formatter.format(maxScore, minScore, finalScore))
```

五、运行效果示例

```
请输入评委人数：5
请输入第1个评委的分数：89
请输入第2个评委的分数：80
请输入第3个评委的分数：98
请输入第4个评委的分数：71
请输入第5个评委的分数：68
去掉一个最高分:98.0
去掉一个最低分:68.0
最后得分:80.0
```

5 列表实验

——整数的因式分解

一、实验目的

（1）掌握列表的创建；

（2）掌握列表的访问；

（3）掌握列表的运算；

（4）了解整数的因式分解方法；

二、实验内容

编写程序，实现整数的因式分解。用户从键盘输入小于 1000 的整数，对其进行因式分解。例如，20=2×2×5, 30=2×3×5。结果保存在列表中，并输出因式分解式。

三、算法分析

对于一个整数 x，从 i（i 初始值为 2）开始判断，如果 x 能被 2 整除，则认为 2 是一个有效因子，将其保存在列表中，然后 x 的值变为 x//2，如果不能被 2 整除，则开始从下一个数 i+1 判断。如此反复，直到最后 x 值变为 1，保存在列表中的所有数相乘就是因式分解的结果。

三、实验步骤

1.运行 jupyter lab

（1）在“实验”文件夹中空白处单击鼠标右键，选择“打开终端”，在终端界面输入“jupyter lab”。

（2）在 jupyter lab 编程界面选择 Notebook 菜单下的 Python3，进入程序编辑器。

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为所做实验名。

2.输入整数 x；

参考代码:

```
x=input("Please Input an integer less than 1000:")
x=eval(x)
t=x
i=2
```

3.创建用于存放因子的列表;

参考代码:

```
#创建空列表，用于保存因子
result=[]
```

4.计算 x 的所有因子，并保存在列表中;

参考代码:

```
#这个 while 循环用来进行因式分解
while t>1:
    if t%i==0:
        result.append(i)
        t=t//i
    else:
        i=i+1
```

5.显示因式分解结果。

参考代码:

```
#显示因式分解的结果
print(x,'=',end='')
for i in range(len(result)):
    if i==len(result)-1:
        print(result[i],end='')
    else:
        print(result[i],'*',end='')
```

五、运行效果示例

```
Please Input an integer less than 1000:80
80 =2 *2 *2 *2 *5
```

AdaBoost 电影数据集数据分类

一、实验目的

（1）掌握 Adaboost 算法基本工作原理；

（2）掌握 pandas、time 中的模块操作方法；

（3）熟悉 sklearn 模块的使用与操作方法；

（5）掌握应用 sklearn 模块实现 Adaboost 数据分类的方法。

二、实验内容

本实验要求通过 AdaBoost 算法将电影数据集（1 train_binary.csv）进行数据分类。首先读取电影数据集中，再将数据集划分为训练集与测试集，然后将数据集进行分类和训练，最后利用测试集数据进行预测分类，获取分类准确率，并显示。

三、实验原理

1.Adaboost 算法概述

Adaboost (adaptive boosting) 是一种迭代算法，其核心思想是针对同一个训练集训练不同的分类器(弱分类器)，然后把这些弱分类器集合起来，构成一个更强的最终分类器(强分类器)。

Adaboost 算法是一种有效而实用的 Boosting 算法。该算法是 Freund 和 Schapire 于 1995 年对 Boosting 算法的改进得到的，其算法原理是通过调整样本权重和弱分类器权值，从训练出的弱分类器中筛选出权值系数最小的弱分类器，组合成一个最终强分类器。基于训练集训练弱分类器，每次下一个弱分类器都是在样本的不同权值集上训练获得的。每个样本被分类的难易度决定权重，而分类的难易度是经过前面步骤中的分类器的输出估计得到的。

Adaboost 算法在样本训练集使用过程中，对其中的关键分类特征集进行多次挑选，逐步训练分量弱分类器，用适当的阈值选择最佳弱分类器，最后将每次迭代训练选出的最佳弱分类器构建为强分类器。其中，级联分类器的设计模式为在尽量保证感兴趣图像输出率的同时，减少非感兴趣图像的输出率，随着迭代次数不断增加，所有的非感兴趣图像样本都不能通过，而感兴趣样本始终保持尽可能通过为止。

Aadboost 算法系统具有较高的检测速率，且不易出现过适应现象。但是该算法在实现过程中为取得更高的检测精度需要较大的训练样本集，在每次迭代过程中，训练一个弱分类器则对应该样本集中的每一个样本，每个样本具有很多特征，因此，从庞大的特征中训练得到最优弱分类器的计算量增大。

2.Adaboost 算法原理

（1）初始化训练数据（每个样本）的权值分布:如果有 N 个样本，则每一个训练的样本点最开始时都被赋予相同的权重：1/N。

（2）训练弱分类器。具体训练过程中，如果某个样本已经被准确地分类，那么在构造下一个训练集中，它的权重就被降低；相反，如果某个样本点没有被准确地分类，那么它的权重就得到提高。同时，得到弱分类器对应的话语权。然后，更新权值后的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。

（3）将各个训练得到的弱分类器组合成强分类器。各个弱分类器的训练过程结束后，分类误差率小的弱分类器的话语权较大，其在最终的分类函数中起着较大的决定作用，而分类误差率大的弱分类器的话语权较小，其在最终的分类函数中起着较小的决定作用。换言之，误差率低的弱分类器在最终分类器中占的比例较大，反之较小。

3.Adaboost 算法流程

（1）计算样本权重

赋予训练集中每个样本一个权重，构成权重向量 D，将权重向量 D 初始化相等值。

假设有 n 个样本的训练集$\{(x_1,y_1),(x_2,y_2),...,(x_n,y_n)\}$，则设定每个样本的权重都相等，则权重为 1/n。

（2）训练并计算错误率

在训练集上训练出一个弱分类器，并计算分类器的错误率。

$$\varepsilon = \frac{N_{error}}{N_{All}}$$

（3）计算弱分类器权重

为当前分类器赋予权重值α，则α计算公式为：

$$\alpha = \frac{1}{2}\ln(\frac{1-\varepsilon}{\varepsilon})$$

（4）调整权重值

根据上一次训练结果，调整权重向量 D(上一次正确分类的权重降低，错误分类的权重增加)。

如果第 *i* 个样本被正确分类，则该样本权重更改为：

$$D_i^{(t+1)} = \frac{D_i^{(t)}e^{-\alpha}}{sum(D)}$$

如果第 i 个样本被错误分类，则该样本权重更改为：

$$D_i^{(t+1)} = \frac{D_i^{(t)}e^{\alpha}}{sum(D)}$$

计算出权重向量 *D* 之后，进入下一次迭代，直到训练错误率或者迭代次数达到用户指定值时停止。

4.相关函数

（1） train_features, test_features, train_labels, test_labels = train_test_split(features, labels, test_size=0.33,random_state=0)

功能：用来随机划分样本数据为训练集和测试集，返回划分好的训练集和测试集标签；

参数 features：所要划分的样本特征集

参数 labels：所要划分的样本结果

参数 test_size：样本占比，如果是整数，就是样本的数量

参数 random_state：随机数的种子。随机数种子：其实就是该组随机数的编号，在需要重复试验的时候，保证得到一组一样的随机数。比如每次都填 1，其他参数一样的情况下，系统得到的随机数组是一样的。但填 0 或不填，每次都会不一样。

（2）AdaBoostClassifier(n_estimators=100, algorithm='SAMME.R')

功能：AdaBoost 分类器

参数 n_estimators：表示要组合的弱分类器个数；

参数 Algorithm：可选{‘SAMME’, ‘SAMME.R’}，默认为‘SAMME.R’，表示使用的是 real boosting 算法，‘SAMME’表示使用的是 discrete boosting 算法。

（3）clf.fit(train_features, train_labels)

功能：在数据集上训练模型。

（4）test_predict = clf.predict(test_features)

功能：预测数据集 test_features 的结果。

（5）accuracy_score(y_true, y_pred, normalize=True, sample_weight=None)

功能：分类正确率分数，返回一个分数，表示正确的比例或样本数；

参数 normalize：默认值为 True，返回正确分类的比例；如果为 False，返回正确分类的样本数。

四、实验步骤

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

2.导入库文件及函数

首先引入 pandas 的相关库，利用库中的对象与函数用于数据分析等。再引入 sklearm 的相关库，利用库中的对象与函数完成数据分类。

参考代码:

```
#1. 导入库文件及函数
import pandas as pd #读取 c s v 文档的
import time #统计时间
from sklearn. model_selection import train_test_split #对读取的数据进行训练集与测试集划分
from sklearn. metrics import accuracy_score#精度得分
from sklearn. ensemble import AdaBoostClassifier#采用提升分类器进行分类
```

3.读取电影数据集

通过 pd.read_csv 函数将 csv 文件读取到 pandas DataFrame 中；通过 train_test_split 函数将原始数据按照比例分割为“测试集”和“训练集”。

参考代码:

```
print("Start read data...")
time_1 = time. time()#记录开始的时间点
raw_data = pd. read_csv('/home/retoo/Desktop/实验/数据集/2. 机器学习/1 train_binary. csv',
header=0)#读取数据
data = raw_data. values#读取到数据值
features = data[::, 1::]#第二列以后为特征数据本身
labels = data[::, 0]#第一列为标签
# 随机选取 33%数据作为测试集，剩余为训练集
train_features, test_features, train_labels, test_labels = train_test_split(features, labels,
test_size=0. 33, random_state=0)
time_2 = time. time()
print('read data cost %f seconds' % (time_2 - time_1))
```

运行结果:

```
Start read data...
read data cost 12.932308 seconds
```

4.AdaBoostClassifier 分类、训练

通过 AdaBoostClassifier( )和 clf.fit(train_features, train_labels)函数将训练集进行分类和训练。

※备注：数据集进行训练，需十分钟左右。

参考代码:

```
print('Start training...')
clf = AdaBoostClassifier(n_estimators=100, algorithm='SAMME. R')
```

```
clf. fit(train_features, train_labels)
time_3 = time. time()
print('training cost %f seconds' % (time_3 - time_2))
```

运行结果:

```
Start training...
training cost 109.770915 seconds
```

5.AdaBoostClassifier 测试

利用 clf.predict(test_features)函数对测试集进行预测。

参考代码:

```
print('Start predicting...')
test_predict = clf. predict(test_features)
time_4 = time. time()
print('predicting cost %f seconds' % (time_4 - time_3))
```

运行结果:

```
Start predicting...
predicting cost 5.906471 seconds
```

6.获取分类准确率分数，并显示

通过 accuracy_score(test_labels, test_predict)函数获取分类准确率分数，并显示。

参考代码:

```
score = accuracy_score(test_labels, test_predict)
print("The accruacy score is %f" % score)
```

运行结果:

```
The accruacy score is 0.988167
```

五、实验结论

本实验利用 Adaboost 算法对电影数据集进行分类，最终的准确率达到%。Adaboost 是一种迭代算法，其核心思想是针对同一个训练集训练不同的分类器(弱分类器)，然后把这些弱分类器集合起来，构成一个更强的最终分类器（强分类器）。

基于 EM 推理的双硬币抛掷模型验证

一、实验目的

（1）掌握 EM 算法的基本原理；

（2）掌握双硬币抛掷模型；

（3）掌握 numpy 库中相关函数、对象的使用方法；

（4）熟练应用 scipy 库中的 binom.pmf()函数计算二项分布概率的方法。

二、实验内容

基于样本数据的模型参数估计是机器学习在大数据分析与处理最常用的无监督学习方法。

本实验要求在掌握双硬币抛掷模型和 EM 算法的基础上，验证基于 EM 算法的双硬币抛掷模型。首先构建双硬币抛掷观测数据集，初始化 A、B 正面朝上的概率；通过 EM 算法迭代计算新的概率，直到参数变化小于阈值或达到最大迭代次数为止。

三、实验原理

在数据分析实践中，常需要从样本观察数据中寻找出样本的模型参数。而最常用的方法就是极大化模型分布的对数似然函数。但是在一些情况下，获得的观察数据有未观察到的隐含数据，即存在未知的有隐含数据和模型参数，因而无法直接用极大化对数似然函数得到模型分布的参数。此时，采用 EM 算法即可有效解决上述问题。

EM 算法思路是使用启发式的迭代方法，既然无法直接求出模型分布参数，那么可以先猜想隐含数据（EM 算法的 E 步），接着基于观察数据和猜测的隐含数据一起来极大化对数似然，求解模型参数（EM 算法的 M 步)。由于之前的隐藏数据是猜测的，所以此时得到的模型参数一般还不是想要的结果。接着，基于当前得到的模型参数，继续猜测隐含数据（EM 算法的 E 步），然后继续极大化对数似然，求解模型参数（EM 算法的 M 步)。以此类推，不断的迭代下去，直到模型分布参数基本无变化，算法收敛，找到合适的模型参数。

从上面的描述可以看出，EM 算法是迭代求解最大值的算法，同时算法在每一次迭代时分为两步，E 步和 M 步。一轮轮迭代更新隐含数据和模型分布参数，直到收敛，即得到我们需要的模型参数。

因此，EM(Expectation Mmaximization)是一种迭代算法，用于含有隐变量(Latent Variable) 的概率模型参数的极大似然估计，或极大后验概率估计。 EM 算法由两步组成，求期望的 E 步和求极大的 M 步。

1.EM 算法原理

E 步：求期望。利用当前估计的参数值来计算对数似然的期望值。

M 步：求极大（后验概率或者极大似然估计）寻找能使 E 步产生的似然期望最大的参数值，然后不断迭代。

EM 算法可以看成是特殊情况下计算极大似然的一种算法。

EM 算法已经有很多应用，比如最经典的 Hidden Markov 模型等。经济学中，除了逐渐开始受到重视的 HMM 模型（例如 Yin and Zhao, 2015），其他领域也有可能涉及到 EM 算法，比如在 Train 的《Discrete Choice Methods with Simulation》就给出了一个 mixed logit 模型的 EM 算法。

本实验利用 EM 算法解决双硬币模型。

2.双硬币抛掷模型

假设有两枚硬币 A、B，以相同的概率随机选择一个硬币，进行如下的抛硬币实验：共做 5 次实验，每次实验独立的抛 10 次，结果如图中 a 所示，例如某次实验产生了 H、T、T、T、H、H、T、H、T、H，H 代表正面朝上。并假设试验数据记录员可能是实习生，业务不一定熟悉，造成 a 和 b 两种情况：

a 表示实习生记录了详细的试验数据，可以观测到试验数据中每次选择的是 A 还是 B；

b 表示实习生忘了记录每次试验选择的是 A 还是 B，无法观测实验数据中选择的硬币是哪个；

试问在 a、b 两种情况下如何估计两个硬币正面出现的概率？

针对 b 实习生的问题，其实和三硬币问题类似，只是这里把三硬币中第一个抛硬币的选择换成了实习生的选择。

对于已知是 A 硬币还是 B 硬币抛出的结果的时候，可以直接采用概率的求法来进行求解。对于含有隐变量的情况，也就是不知道到底是 A 硬币抛出的结果，还是 B 硬币抛出的结果的时候，就需要采用 EM 算法进行求解了，如图 1 所示。

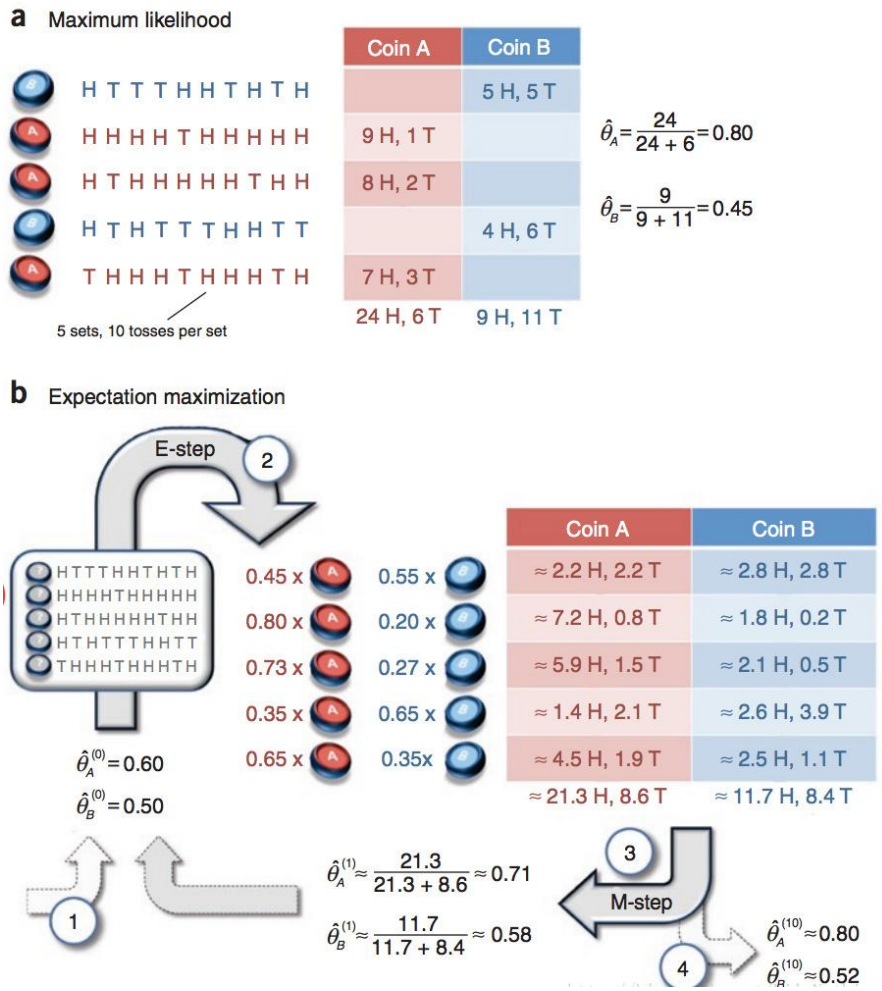


图 1　EM 算法求解示意图

EM 算法的第一步就是初始化模型参数，再利用当前估计的参数值来计算对数似然的期望值（E 步），然后寻找能使 E 步产生最大似然期望的参数值（M 步），重复 E、M 步，不断迭代，直到模型参数变化小于阈值或达到最大迭代次数。

3.相关函数

（1）binom.pmf(k,n,p)

功能:计算每次成功概率为 p 的情况下，做某件事 n 次，k 次成功的二项分布概率。

参数 n: 做某件事 n 次；

参数 p: 每次成功的概率为 p；

参数 k:成功 k 次；

返回值：k 次成功的二项分布概率

四、实验步骤

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

2.导入库文件及函数

首先引入 numpy 库，利用 numpy 库中的对象与函数存储和处理数据等；再引入 scipy.stats 的相关库，利用库中的对象与函数求解二项分布概率等。

参考代码:

```
#1. 导入库文件及函数
import numpy
from scipy. stats import binom
```

3.构建观测数据集，初始化双硬币抛掷模型参数

采集数据，构建观测数据集：用表示（正面），表示（反面）；初始化双硬币模型参数：硬币正面朝上的概率为.，硬币正面朝上的概率为.。

参考代码:

```
observations = numpy. array([[1, 0, 0, 0, 1, 1, 0, 1, 0, 1],
                             [1, 1, 1, 1, 0, 1, 1, 1, 1, 1],
                             [1, 0, 1, 1, 1, 1, 1, 0, 1, 1],
                             [1, 0, 1, 0, 0, 0, 1, 1, 0, 0],
                             [0, 1, 1, 1, 0, 1, 1, 1, 0, 1]])
priors=[0. 6, 0. 5]
```

4.编写双硬币抛掷模型迭代函数

函数参数 priors：初始化参数 theta_A，theta_B；observations：观测矩阵 M*N（M 代表实验组数，N：代表每组实验的次数）。返回：新参数[new_theta_A,new_theta_B]。

（1）E 步：计算当前参数下 A、B 硬币产生的正反面次数。

抛硬币是一个二项分布，可以用 scipy 中的 binom.pmf 来计算。

contribution_A = binom.pmf(num_heads,len_observation,theta_A)

contribution_B = binom.pmf(num_heads,len_observation,theta_B)

将两个概率正规化，得到数据来自硬币 A，B 的概率：

weight_A = contribution_A / (contribution_A + contribution_B)

weight_B = contribution_B / (contribution_A + contribution_B)

通过 weight_A,weight_B 可以估计数据中 A、B 分别产生正反面的次数了。weight_A 代表数据来自硬币 A 的概率的估计，将它乘上正面的总数，得到正面来自硬币 A 的总数，同理可得 A 的反面、B 的正反面的总数。

更新在当前参数下 A、B 硬币产生的正反面次数

counts['A']['H'] += weight_A * num_heads

counts['A']['T'] += weight_A * num_tails

counts['B']['H'] += weight_B * num_heads

counts['B']['T'] += weight_B * num_tails

（2）M 步：当前模型参数下，计算新的模型参数。

new_theta_A = counts['A']['H']/(counts['A']['H'] + counts['A']['T'])

new_theta_B = counts['B']['H']/(counts['B']['H'] + counts['B']['T'])

参考代码:

```
#迭代函数
def em_single(priors, observations):
    counts={'A':{'H':0,'T':0},'B':{'H':0,'T':0}}
    theta_A=priors[0]
    theta_B=priors[1]
    # E 步：计算当前参数下 A、B 硬币产生的正反面次数。
    for observation in observations:
        len_observation = len(observation)
        num_heads = observation.sum()#正面次数
        num_tails = len_observation-num_heads#反面次数
        #二项分布求解公式,抛硬币是一个二项分布
        contribution_A = binom.pmf(num_heads, len_observation, theta_A)
        contribution_B = binom.pmf(num_heads, len_observation, theta_B)
        #将两个概率正规化，得到数据来自硬币 A，B 的概率。
        weight_A = contribution_A / (contribution_A + contribution_B)
        weight_B = contribution_B / (contribution_A + contribution_B)
        #更新在当前参数下 A，B 硬币产生的正反面次数
        counts['A']['H'] += weight_A * num_heads
        counts['A']['T'] += weight_A * num_tails
        counts['B']['H'] += weight_B * num_heads
        counts['B']['T'] += weight_B * num_tails
    # M 步：计算新的模型参数
    new_theta_A = counts['A']['H'] / (counts['A']['H'] + counts['A']['T'])
    new_theta_B = counts['B']['H'] / (counts['B']['H'] + counts['B']['T'])
    return [new_theta_A, new_theta_B]
```

5.编写算法主循环函数

参数：观测数据；：模型初值；：迭代结束阈值；：最大迭代次数；返回：最终的权重及迭代次数。算法的主循环的两个终止条件：模型参数变化小于阈值；循环达到最大次数。

参考代码:

```
#EM 算法主循环函数
def em(observations, prior, threshold = 1e-6, iterations=10000):
    iteration = 0
    while iteration <iterations:
        new_prior = em_single(prior, observations)
        delta_change = numpy.abs(prior[0]-new_prior[0])
        if delta_change < threshold:
            break
        else:
            prior = new_prior
            iteration +=1
        #返回最终的权重，以及迭代次数
    return new_prior, iteration
```

6.调用算法迭代函数和循环函数，输出迭代结果

调用第、步骤函数，进行算法迭代，并输出结果。

参考代码:

```
new_thetas=em_single(priors, observations)
print (new_thetas)

new_thetas, iteration=em(observations, priors)
print(new_thetas)
print(iteration)
```

运行结果：

```
[0.7130122354005162, 0.5813393083136627]
[0.7967887593831098, 0.5195839356752803]
14
```

五、实验结论

本实验利用 EM 算法推理(a)，(b)两种情况下，A,B 两张硬币正面朝上的概率，当只迭代一次是 A 正面朝上的概率是 0.71，B 正面朝上的概率是 0.58，当迭代次数为 14

次时，A 正面朝上的概率是 0.79，B 正面朝上的概率是 0.51，本实验中所用到的 EM（Expectation-Maximization）算法可以对结果进行预测参数的信息，在一些含有隐变量的参数中使用它可以对模型参数进行一定的估计，并且它在机器学习的很多领域都会运用。

基于 K 均值算法的未知数据分类

一、实验目的

（1）掌握 k 均值聚类的基本原理与实现方法；

（2）掌握随机数据生成方法；

（3）掌握 matplotlib.pyplot 库中基本绘图函数的使用方法。

二、实验内容

聚类分析是机器学习无监督学习的常用方法之一，通过对数据相似度进行计算，完成数据的多元分类任务。

本实要求在掌握 K-Means 聚类算法原理的基础上，对一组随机二维坐标数据进行分类。通过 numpy 库中 random.randint 函数随机生成二维坐标数据，初始化聚类中心，计算各数据与聚类中心点的距离，迭代计算，实现对所有数据的聚类分析。

三、实验原理

k 均值聚类是使用最大期望算法（Expectation-Maximization algorithm）求解的高斯混合模型（Gaussian Mixture Model，GMM）在正态分布的协方差为单位矩阵。为了更好的理解 *k* 均值聚类，将使用随机生成的 100 个坐标点，进行聚类中心求取。

1. *K*-Means 原理

K-Means 算法的思想很简单，对于给定的样本集，按照样本之间的距离大小，将样本集划分为 *K* 个簇。让簇内的点尽量紧密地连在一起，而让簇间的距离尽量的大。

如果用数据表达式表示，假设簇划分为(C_1,C_2,...C_k)，则目标是最小化平方误差 *E*：

$$E = \sum_{i=1}^{k} \sum_{x \in Ci} \left\| x - \mu_i \right\|_2^2$$

其中μ_i是簇 C_i 的均值向量，有时也称为质心，表达式为：

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

如果想直接求上式的最小值并不容易，这是一个 NP 问题，因此只能采用启发式的迭代方法。

K-Means 采用的启发式方式很简单，用下面一组图就可以形象的描述。

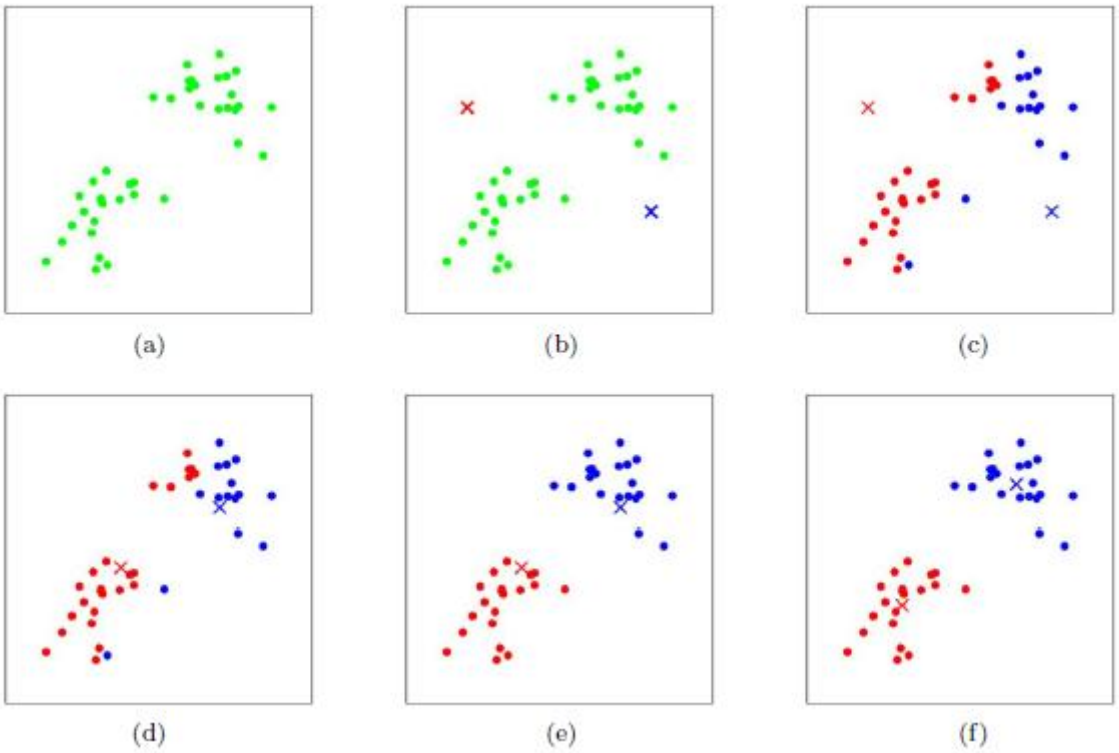


图 1　K-Means 启发方式

上图(a)表达了初始的数据集，假设 k=2。在图(b)中，我们随机选择了两个 k 类所对应的类别质心，即图中的红色质心和蓝色质心，然后分别求样本中所有点到这两个质心的距离，并标记每个样本的类别为和该样本距离最小的质心的类别，如图(c)所示，经过计算样本和红色质心和蓝色质心的距离，得到了所有样本点的第一轮迭代后的类别。此时对当前标记为红色和蓝色的点分别求其新的质心，如图(d)所示，新的红色质心和蓝色质心的位置已经发生了变动。图(e)和图(f)重复了图(c)和图(d)的过程，即将所有点的类别标记为距离最近的质心的类别并求新的质心。最终得到的两个类别如图(f)。

当然在实际 K-Mean 算法中，一般会多次运行图(c)和图(d)，才能达到最终的比较优的类别。

2. K-Means 实验过程

先随机选取 K 个对象作为初始的聚类中心。然后计算每个对象与各个种子聚类中心之间的距离，把每个对象分配给距离它最近的聚类中心。聚类中心以及分配给它们的对象就代表一个聚类。一旦全部对象都被分配了，每个聚类的聚类中心会根据聚类中现有的对象被重新计算。这个过程将不断重复直到满足某个终止条件。终止条件可以是以下任何一个：

（1）没有（或最小数目）对象被重新分配给不同的聚类；

（2）没有（或最小数目）聚类中心再发生变化；

（3）误差平方和局部最小。

其过程如下：

（1）选择 k 个点作为初始质心；

（2）将每个点指派到最近的质心，形成 k 个簇，重新计算每个簇的质心

（3）重复第（2）步，直到质心不发生变化。

四、实验步骤

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

2.导入库文件及函数

首先引入 numpy 库，利用 numpy 库中的相关对象与函数存储和处理数据等；导入绘图库 matplotlib.pyplot，用于形象地表达出聚类后的元素显示（散点图）。

参考代码:

```
import numpy as np
import matplotlib. pyplot as plt
```

3.编写计算两点间距离、计算质心、寻找距离质心最远元素函数

函数说明:

（1）def distance(e1, e2):求解两点之间的距离。输入为带坐标信息的元组值 e1、e2，输出为两点之间的浮点型距离。

（2）def means(arr)：求取一组列表数据的集合中心。输入为列表，输出为存放同类型元素的多维数组。

（3）def farthest(k_arr, arr)：找出 arr 数组中，距离 k_arr 最远的元素，用于初始化聚类中心。输入 k_arr, arr 为 numpy.ndarray，输出 f 为 list。

参考代码:

```
def distance(e1, e2):
    return np.sqrt((e1[0]-e2[0])**2+(e1[1]-e2[1])**2)

def means(arr):
    return np.array([np.mean([e[0] for e in arr]), np.mean([e[1] for e in arr])])

def farthest(k_arr, arr):
    f = [0, 0]
```

```
    max_d = 0
    for e in arr:
        d = 0
        for i in range(k_arr.__len__()):
            d = d + np.sqrt(distance(k_arr[i], e))
        if d > max_d:
            max_d = d
            f = e
    return f
```

4.随机生成数据

使用 np.random.randint 生成二维随机坐标。若此处如果有数据集就不需要随机生成。

参考代码:

```
arr = np.random.randint(100, size=(100, 1, 2))[:, 0, :]
```

5.初始化聚类中心和聚类容器

首先随机选择集合里的一个元素作为第一个聚类中心放入容器，选择距离第一个聚类中心最远的一个元素作为第二个聚类中心放入容器，第三、四、...、N 个同理，为了优化可以选择距离开方做为评判标准。

参考代码:

```
m = 5
r = np.random.randint(arr.__len__() - 1)
k_arr = np.array([arr[r]])
cla_arr = [[]]
for i in range(m - 1):
    k = farthest(k_arr, arr)
    k_arr = np.concatenate([k_arr, np.array([k])])
    cla_arr.append([])
```

6.迭代聚类

依次把集合里的元素与距离最近的聚类中心分为一类，放到对应该聚类中心的新的容器，一次聚类完成后求出新容器里个类的均值，对该类对应的聚类中心进行更新，再次进行聚类操作，迭代 n 次得到理想的结果。

参考代码:

```
n = 20
cla_temp = cla_arr
for i in range(n):      # 迭代 n 次
    for e in arr:       # 把集合里每一个元素聚到最近的类
```

```
            ki = 0          # 假定距离第一个中心最近
            min_d = distance(e, k_arr[ki])
            for j in range(1, k_arr.__len__()):
                if distance(e, k_arr[j]) < min_d:      # 找到更近的聚类中心
                    min_d = distance(e, k_arr[j])
                    ki = j
            cla_temp[ki].append(e)
        # 迭代更新聚类中心
        for k in range(k_arr.__len__()):
            if n - 1 == i:
                break
            k_arr[k] = means(cla_temp[k])
            cla_temp[k] = []
```

7.可视化显示分类结果

利用 python 第三方库中的可视化工具 matplotlib.pyplot 对聚类后的元素显示（散点图）。

参考代码:

```
col = ['HotPink', 'Aqua', 'Chartreuse', 'yellow', 'LightSalmon']
for i in range(m):
    plt.scatter(k_arr[i][0], k_arr[i][1], linewidth=10, color=col[i])
    plt.scatter([e[0] for e in cla_temp[i]], [e[1] for e in cla_temp[i]], color=col[i])
plt.show()
```

运行结果：

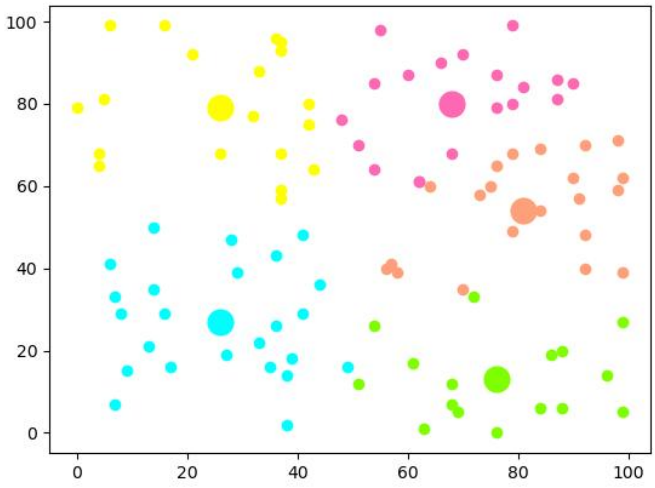


图 2　迭代聚类效果图

五、实验结论

本实验利用 K-means 对数据进行聚类，K-Means 是个简单实用的聚类算法，通过该

实验帮助同学更好的理解无监督学习技术的常用聚类方法，此外 K-Means 具有原理比较简单，实现也是很容易，收敛速度快，聚类效果较优等优势，但它并不完美，需要多次运行该算法才能避免次优解，并且还需要指定集群数，除了无监督学习中会用到聚类算法，其实在有监督和无监督学习都会应用到聚类的算法，比如要分析卫星图像以测量某个区域中有多少森林总面积就可以应用聚类来分割。

1　线性回归建模与训练

一、实验目的

（1）掌握线性回归的概念与基础；

（2）掌握线性回归的激活函数、损失函数、输出层、优化函数；

（3）掌握利用梯度下降法求解最优值的方法；

（4）了解 PyTorch 下相关概念与函数；

（5）掌握深度学习准备数据集、建模、训练的流程。

二、实验内容

本实验要求在掌握线性回归基本原理的基础上，初识深度学习准备数据集、建模、训练的流程。首先随机生成数据集；接着构建线性回归模型，随机抽取小批量训练数据样本进行训练，采用随机梯度下降法迭代寻找最优的权重参数；然后比较权重的真实值与训练值，若相差很小，即获得线性回归模型的参数，否则手动修改迭代次数，重新迭代。

三、实验原理

回归问题与分类问题不同，回归问题的预测值是一系列的连续值，而分类问题的预测值则是一系列的离散值。回归问题在生活中最常见的就是针对房屋价格的预测、气温的预测、销售额的预测；分类问题则主要表现为图像的分类、垃圾邮件的分类、病毒的检测等等。Logistic（线性）回归名称上虽然也称之为回归，但其本质上是分类问题。

1.线性回归的单层神经网络

为了更加清晰展示线性回归结构在神经网络中的表现形式，图 1 利用神经网络图对线性回归模型进行展示。

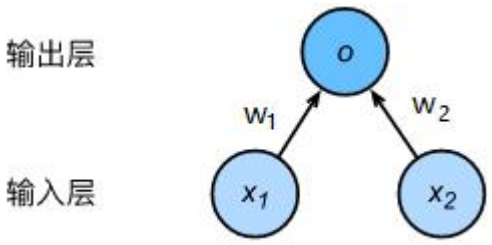


图 1 单层神经网络模型

以房屋价格预测为例，房屋价格的影响因素有很多，如:房屋的房龄、房屋的地段、房屋的市场行情等等。假设只考虑两个方面的因素，房屋的地段和房屋的年龄，接下来针对这两个因素判断其与房屋的价格之间的关系。

2.模型

假设房屋的地段为x_1，房屋的房龄为x_2，房屋的价格为y，因此可以建立基于输入x_1和x_2来计算y之间的关系，这就是线性回归模型，输入和输出之间的线性关系为：

$$\hat{y} = w_1 x_1 + w_2 x_2 + b$$

其中w_1和w_2是权重（weight），b是偏置（bias），且均为标量，以上参数则为线性回归模型的参数（parameter）。模型输出$\hat{y}$是线性回归对真实价格 y 的预测或估计。在实际使用时通常允许二者之间有一定误差。

3.训练数据

通常的情况下，需要收集一系列的的真实数据，比如真实的房屋价格和它所对应的地段和房龄。接着基于收集到的数据训练模型，使得模型的预测价格和真实价格尽可能地接近。这个数据集可以分为训练集和测试集，一条房屋数据称为一个样本，真实的房屋价格成为了标签，房屋地段和房屋房龄称为特征。假设采集的样本数为 num，索引为i的样本的特征为x_1^i和x_2^i，标签为y^i。对于索引为 i 的房屋，线性回归模型的房屋价格预测表达式为

$$\hat{y}^i = w_1 x_1^i + w_2 x_2^i + b$$

4.损失函数

损失函数用来衡量真实值和预测值之间的差距，数值越小代表真实值和预测值越接近，常用的线性回归模型评估函数是平方函数。

$$L^i(\hat{y}, y) = \frac{1}{2}(\hat{y} - y^i)^2$$

上面的公式称为平方损失函数，在模型的训练过程中，目的是找到一组w_1^*、w_2^*以及b来使得损失函数最小。

$$Loss = \arg\min L^i(\hat{y}, y)$$

5.随机梯度下降

通过随机挑选一组模型的初始值，在优化的过程中对选择的参数进行多次迭代，每次迭代的过程中都可能降低损失函数的值。在迭代的过程中，随机均匀采样一个由固定数目训练数据样本所组成的小批量，求小批量中训练数据样本的平均损失的导数（梯度），最后与预先设定的一个正数的乘积作为模型参数在本次迭代的减小量。

梯度下降算法是一种常用的数值求解函数最小值的基本方法，它的基本思想就像盲人下山。这里要优化的损失函数$L(w_1, w_2, b)$就是那座山。假设有一个盲人站在山上的某个随机初始点（这里对应了 a 和 b 的初始随机值），他会在原地转一圈，寻找最快的方向来行进。所谓下降的快慢，其实就是 L 对 w_1、w_2、b 在这一点的梯度（导数）；所谓的行进，就是更新 a 和 b 的值，让盲人移动到一个新的点。于是，每到一个新的点，盲人就会依照同样的方法行进，最终停留在让 L 最小的那个点。

可以通过以下公式迭代计算来实现盲人下山的过程：

$$w_1^{t+1} = w_1^t - \alpha \frac{\partial L}{\partial w_1}\Big|_{w1=w_1^t}$$

$$w_2^{t+1} = w_2^t - \alpha \frac{\partial L}{\partial w_2}\Big|_{w_2=w_2^t}$$

$$b_{t+1} = b_t - \alpha \frac{\partial L}{\partial b}\Big|_{b=b_t}$$

α 为一个参数，叫学习率，它可以调节更新的快慢，相当于盲人每一步的步伐有多大。α 越大，w_1，w_2，b 更新越快，但是计算得到的最优值 L 就有可能越不准。

6.模型预测

模型预测就是利用训练好的模型，对训练集以外的数据（如：测试集数据）进行预测，这里利用测试集的房屋地段和房屋房龄数据预测房子的价格。

四、实验步骤

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

2. 导入库文件及相关函数

引入 Torch 相关库，利用 Torch 的相关库中的对象与函数完成数据集的建立；引入 matplotlib 库，用来绘图显示。

参考代码:

```
%matplotlib inline
import torch
from IPython import display
from matplotlib import pylab as plt
import numpy as np
import random
```

3.生成数据集，读取数据集并显示，随机抽取抽取小批量数据样本

（1）生成数据集

深度学习最重要的就是数据的建立，本实验作为深度学习的入门级实验，仿照图片

数据格式建立一个简单的人工数据集，完成原始的数据收集。假设训练数据样本数量为100 个，输入特征为 2，线性回归的真实权重为 $w=[3,2.2]$ 偏置 $b=3$，那么生成的真实标签为：

$$y=wx+b$$

参考代码:

```
num_inputs=2
num_sample=100
W=[3,-2.2]
B=3
Xinput=torch.randn(num_sample,num_inputs,dtype=torch.float32)#随机生成样本数据
labels=W[0]*Xinput[:,0]+W[1]*Xinput[:,1]+B
print(Xinput[0],labels[0])
```

模型输入特征和对应的标签输出的结果如下：

```
tensor([-0.0678,  0.7837]) tensor(1.0725)
```

（2）绘制这 2 个特征与对应的标签之间关系

参考代码:

```
def use_svg_display():
    display.set_matplotlib_formats('svg') #设置绘图为矢量图

def set_fig_size(figsize=(4.2,3.4)):
    use_svg_display()
    plt.rcParams['figure.figsize']=figsize#  设置绘图的尺寸

#plt.rcParams['font.sans-serif']=['SimHei']# kentchen font.san-serif
plt.rcParams['axes.unicode_minus']=False
set_fig_size()
plt.scatter(Xinput[:,1].numpy(),labels.numpy(),1)
plt.show()
print(Xinput.shape,labels.shape)
```

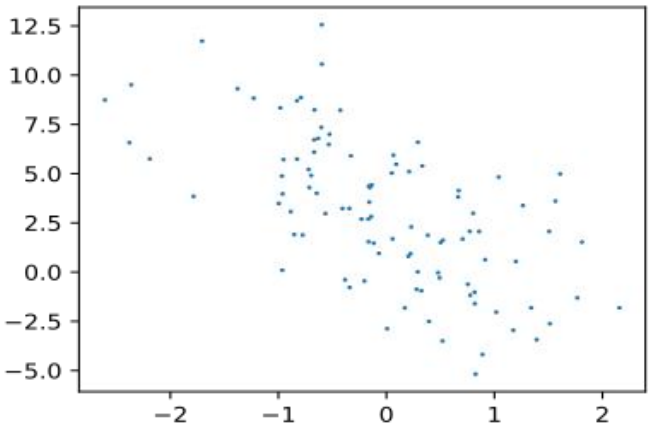


图 2　人工数据集分布图

（3）定义随机抽取小批量训练数据样本函数

参考代码:

```
batch_size=2
def data_iter(batch_size, Xinput, labels):
    num_Xinput = len(Xinput)
    indices = list(range(num_Xinput))
    random.shuffle(indices)  # 样本的读取顺序是随机的
    for i in range(0, num_Xinput, batch_size):
        j = torch.LongTensor(indices[i: min(i + batch_size, num_Xinput)]) # 最后一次可能不足一个
batch
        yield  Xinput.index_select(0, j), labels.index_select(0, j)
```

4.定义线性回归模型

设置参数初始值，定义线性回归模型。其中模型的权重初始化为均值为 0、标准差为 0.01 的一个正态随机数，偏置则初始化为 0，训练模式下模型参数的梯度设为可训练。

参考代码:

```
w = torch.tensor(np.random.normal(0, 0.01, (num_inputs, 1)),
dtype=torch.float32).requires_grad_(requires_grad=True)
b = torch.zeros(1, dtype=torch.float32).requires_grad_(requires_grad=True)

def linreg(X, w, b):
    return torch.mm(X, w) + b
```

5.定义损失函数

定义损失函数，用来评估真实值和预测值之间的距离，这里采用平方损失函数。

参考代码:

```
def squared_loss(y_hat, y):
    # 注意这里返回的是向量, 另外, pytorch 里的 MSELoss 并没有除以 2
    return (y_hat - y.view(y_hat.size())) ** 2 / 2
```

6.定义优化函数

利用随机梯度下降算法不断迭代，使得模型参数不断地优化，使预测值不断接近真实值。

参考代码:

```
def sgd(params, lr, batch_size):
    for param in params:
        param.data -= lr * param.grad / batch_size # 注意这里更改 param 时用的 param.data
```

7.训练模型

在训练模型中，一般通过多次迭代模型的参数，来寻找最优的权重值，具体操作是通过当前的小批量样本，以及反向传播函数计算小批量随机梯度，并调用优化函数来迭代模型训练。假设训练 3 个 epoch，网络模型使用的是线性回归模型，损失函数使用的是平方损失函数，并设置学习率为 0.03。更改迭代参数 echochs，修改迭代次数。

（1）训练模型

参考代码:

```
lr = 0.03
epochs = 3
net = linreg
loss = squared_loss
for epoch in range(epochs):  # 训练模型一共需要 epochs 个迭代周期
    # 每一个迭代周期，会使用训练数据集中所有样本一次（假设样本数能够被批量大小整除）。
    # X 和 y 分别是小批量样本的特征和标签
    for X, y in data_iter(batch_size, Xinput, labels):
        Model = net(X, w, b)
        l = loss(Model, y).sum()  # l 是有关小批量 X 和 y 的损失
        l.backward()  # 小批量的损失对模型参数求梯度
        sgd([w, b], lr, batch_size)  # 使用小批量随机梯度下降法迭代模型参数

        # 梯度清零
        w.grad.data.zero_()
        b.grad.data.zero_()
    train_l = loss(net(Xinput, w, b), labels)
    print('epoch %d, loss %f' % (epoch + 1, train_l.mean().item()))
```

通过 3 个 epoch 后的模型训练结果展示如下:

```
epoch 1, loss 0.531965
poch 2, loss 0.026290
epoch 3, loss 0.001344
```

由训练损失值可知，该损失值在不断地下降，说明模型网络正处于优化状态。

（2）显示参数权重的训练值与真实值。

训练结束后，显示权重的训练值与真实值，比较真实的权重值和训练的权重值是否相差很小。

参考代码:

```
print(W, '\n', w)
print(B, '\n', b)
```

打印结果如下:

```
[3, -2.2]
 tensor([[ 2.8910],
         [-2.1671]], requires_grad=True)
3
 tensor([2.9190], requires_grad=True)
```

由打印的结果可知，该模型参数权重的训练值与真实值相差很小，对生成数据的拟合比较准。

五、实验结论

本实验通过线性回归算法拟合模型参数，用 MSE 函数作损失函数优化结果，最终输出权重为[2.8910,-2.1671]与实际值[3,-2.2]十分接近，输出的偏置值 2.9190 与实际值 3 十分接近，拟合效果理想。通过本实验，能够帮助我们理解一个简单的神经网络的大致工作原理，利用损失函数的对模型进行优化，求出最优的参数组合，这是深度学习中重要的环节。

2　基于神经网络的服装分类

一、实验目的

（1）掌握神经网络的激活函数、损失函数、输出层、最优化；

（2）掌握多层神经网络的结构、原理；

（3）了解 PyTorch 下相关概念与函数；

（4）了解服装分类。

二、实验内容

本实验要求在掌握多层神经网络结构、原理的基础上，实现基于神经网络的服装分类。通过 Pytorch 建立神经三层网络模型，利用服装数据集 Fashion-Mnist 数据进行训练，采用梯度下降法对模型进行优化；最后对模型进行训练，获取模型在训练集和测试集上的准确率和损失函数值。

三、实验原理

线性回归是单层神经网络，然而深度学习模型大多是多层神经网络，如常见的多层感知机神经网络。多层感知机神经网络由一层或者多层隐藏层和输入层以及输出层构成，是一个完整的神经网络模型。

1.感知机的多层神经网络

图 1 是含 1 层隐藏层的多层感知机

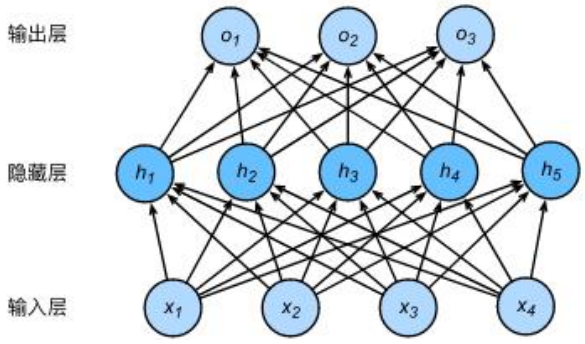


图 1　多层感知机模型

图 1 所示的多层感知机，输入层有 5 个输入神经元，隐藏层有 6 个隐藏单元，输出层有 4 个输出神经元。由图可知，各个输入神经元和各个隐藏单元连接，而每个隐藏单元又和每个输出神经元相互连接。假设输入的样本量大小为 m，输入的个数为 n，且多层感知机只有一个隐藏层，其隐藏单元数量为 h，将隐藏单元的输出标注为 H，隐藏层的权重和偏置记为W_h和b_h,输出层的权重和偏置记为W_o和b_o，首先计算输入层和隐藏层

之间的输出:

$$H = W_h x + b_h$$

接着计算隐藏层和输出层之间的输出:

$$O = W_o H + b_o$$

从计算公式可知，多层感知机多层神经网络与线性回归的单层神经网络相比，虽然前者加入了隐藏层，但是二者的计算方式相似。

2.激活函数

多层感知机和线性回归相似的根本问题在于二者都只是简单地仿射变换，即使再多层的隐藏层也只是简单的叠加仿射，因此需要引入非线性函数来使得隐藏层输出成为非线性变换，这种非线性函数就是激活函数，有了这样的非线性激活函数，神经网络的表达能力更加强大了。每一层的输出通过激活函数，就会变得比以前复杂很多，从而提升神经网络的表达能力，常见的激活函数如下:

（1）Sigmoid 函数

sigmoid 函数也叫 Logistic 函数，用于隐层神经元输出，取值范围为(0,1)，它可以将一个实数映射到(0,1)的区间，可以用来做二分类。在特征相差比较复杂或是相差不是特别大时效果比较好。Sigmoid 作为激活函数有以下优缺点：

优点：平滑、易于求导。

缺点：激活函数计算量大，反向传播求误差梯度时，求导涉及除法；反向传播时，很容易就会出现梯度消失的情况，从而无法完成深层网络的训练。

$$P = \frac{1}{1 + e^{-z}}$$

其中e是纳皮尔常数，其值是 2.7182…

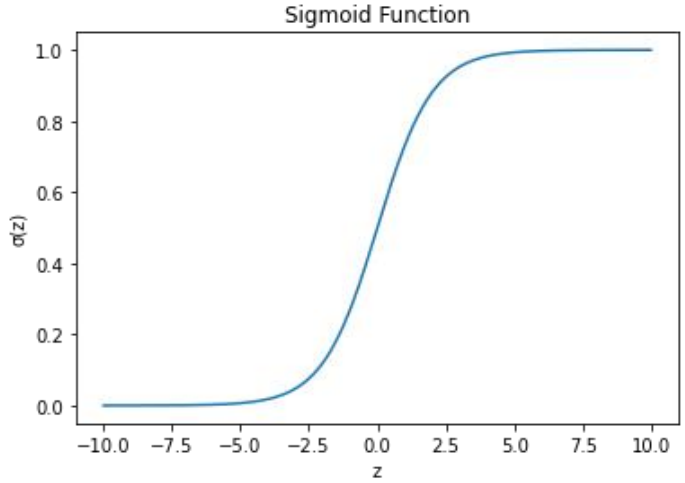


图 2 Sigmoid 函数图像

（2）Tanh 函数

Tanh 函数也称为双切正切函数，取值范围为[-1,1]。Tanh 在特征相差明显时效果很好，在循环过程中不断扩大特征效果，与 sigmoid 的区别是 tanh 是 0 均值的，因此实际应用中 tanh 会比 sigmoid 更好。

$$F(x)=\frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})}$$

tanh 函数与 sigmoid 函数的曲线是比较相近的，输出区间在（-1,1）之间，整个函数以 0 为中心。

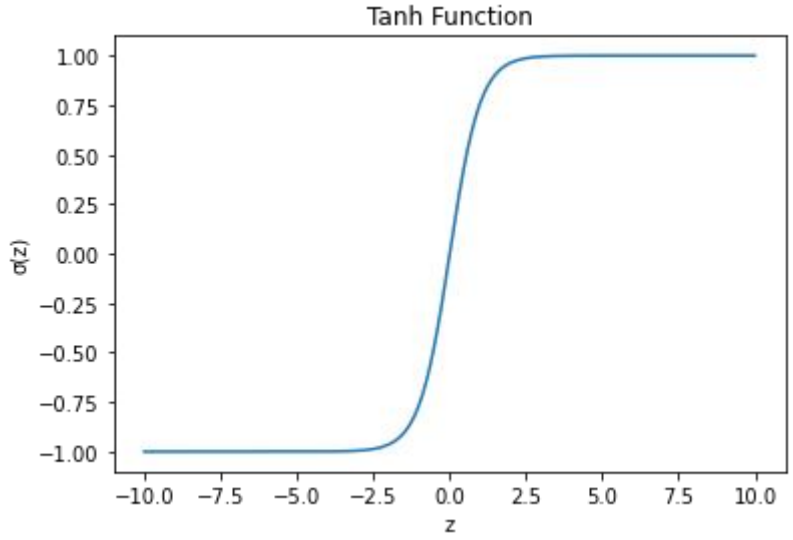


图 3 Tanh 函数图像

（3）ReLU 函数

又称为修正线性单元，是一个分段函数: $F(x)=\max(0,x)$ ，大于 0 的数直接输出，小于 0 的数则输出为 0，在 0 这个地方不连续。其优点是在随机梯度下降法的训练中收敛很快，在输入为正数的时候，不存在梯度饱和问题。

加入激活函数后的多层感知机的特征传递公式为:

$$H=\varphi(W_h x+b_h)$$
$$O=W_o H+b_o$$

相比于传统的神经网络激活函数，诸如逻辑函数（Logistic sigmoid）和 tanh 等双曲函数，ReLU 函数有着以下几方面的优势：

①仿生物学原理。相关大脑方面的研究表明生物神经元的讯息编码通常是比较分散及稀疏的。通常情况下，大脑在同一时间大概只有 1%-4%的神经元处于活跃状态。使用线性修正以及正规化（regularization）可以对机器神经网络中神经元的活跃度（即输出为正值）进行调试；相比之下，逻辑函数在输入为 0 时达到 0.5，即已经是半饱和的稳定状态，不够符合实际生物学对模拟神经网络的期望。不过需要指出的是，一般情况下，在一个使用 ReLU 的神经网络中大概有 50%的神经元处于激活态。

②更加有效率的梯度下降以及反向传播，避免了梯度爆炸和梯度消失问题;

③简化计算过程。没有了其他复杂激活函数中诸如指数函数的影响；同时活跃度的分散性使得神经网络整体计算成本下降。

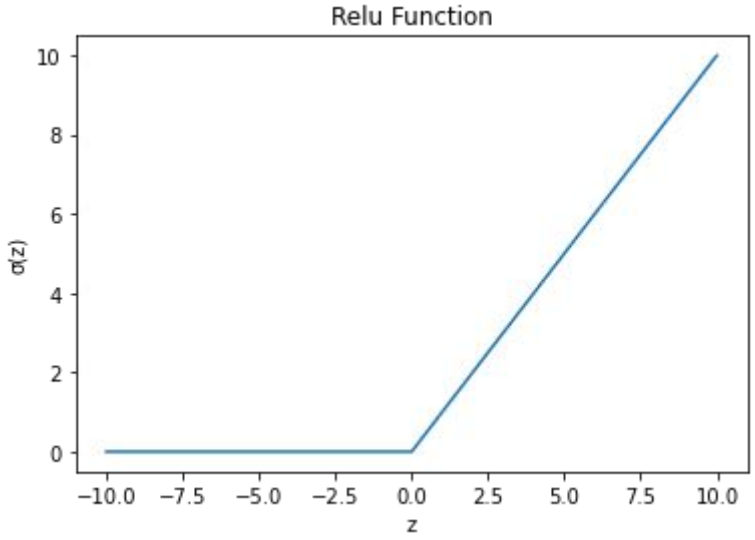


图 4 Relu 函数图像

3.训练数据

用于分类的训练数据有很多，如 Mnist,cifar-10 等，由于本实验不对数据集进行分析，因此可以直接读取该数据集或者自定义符合 Mnist 数据集格式的数据集。希望多层感知机神经网络通过训练集能够训练出对数据类别准确分类的模型，Mnist 数据集总共包含 10 个类别的数据，图片的形状为 28*28，但多层感知机需要将数据转换为 28*28=784 的向量进行训练，因此，网络的输入是 784，输出是 10，在本实验中，这里设计隐藏层的隐藏单元有 256 个。

4.交叉熵损失函数

损失函数在模型训练中用来衡量真实值和预测值之间的差距，数值越小代表真实值和预测值越接近，交叉熵常用来表示衡量两个不同概率分布的差异程度，在深度学习中就是真实概率分布和预测概率分布之间的差异。真实标签可以用类别分布表示，对于真实样本 i，可以构造标签个数的向量空间 Y 。使得 y^i 个元素为 1，其余的为 0，这样，在训练过程中只需要预测值 $\hat{y}^i$ 真实概率尽可能地接近 1 就行。比如在图像分类中，$y^i=3$，那么只需要预测值 $\hat{y}^i$ 比其余的 $\hat{y}^{i-1}$ 和 $\hat{y}^{i+1}$ 都大就可以了，尽管 $\hat{y}^i$ 只有 0.5，那它也是预测正确的，但平方误差就要求严格很多，那么交叉熵是对两个概率分布的衡量差异最合适的函数，它的数学公式如下：

$$H(x)=-\sum p(x_x)\log(p(x_x))$$

5.随机梯度下降

通过随机挑选一组模型的初始值，在优化的过程中对选择的参数进行多次迭代，每次迭代的过程中都可能降低损失函数的值。在迭代的过程中，随机均匀采样一个由固定数目训练数据样本所组成的小批量，求小批量中训练数据样本的平均损失的导数（梯度），最后与预先设定的一个正数的乘积作为模型参数在本次迭代的减小量。

四、实验步骤

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

2.导入库文件及相关函数

引入 Torchvision 相关库，利用 Torchvision 相关模块的对象与函数完成数据集的建立；引入 numpy 库用于科学计算；引入 matplotlib.pyplot 库用于绘图。

参考代码:

```
%matplotlib inline
import torch
import numpy as np
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
```

3.建立数据集

（1）下载 Fashion-Mnist 服装数据集

※备注：下载数据集，需要等待几分钟。

通过 torchvision.datasets.FashionMNIST 函数下载 Fashion-Mnist 数据集，该数据集包括训练集和测试集两个数据集，包含 10 个类别，分别为：t-shirt（T 恤）、trouser（裤子）、pullover（套衫）、dress（连衣裙）、coat（外套）、sandal（凉鞋）、shirt（衬衫）、sneaker（运动鞋）、bag（包）和 ankle boot（短靴），并将其对应为相应的标签，且数据集为灰度图像，因此通道数为 1。

参考代码:

```
batch_size = 256
num_workers = 4
#下载训练集到指定目录
mnist_train = torchvision.datasets.FashionMNIST(root='/home/retoo/Desktop/实验/数据集/3.深度学习/2FashionMNIST', train=True, download=True, transform=transforms.ToTensor())
# 下载测试集到指定目录
mnist_test = torchvision.datasets.FashionMNIST(root='/home/retoo/Desktop/实验/数据集/3.深度学习/2FashionMNIST', train=False, download=True,transform=transforms.ToTensor())
# 将训练集组成一个批次
train_iter = torch.utils.data.DataLoader(mnist_train, batch_size=batch_size, shuffle=True,
num_workers=num_workers)
# 将测试集组成一个批次
test_iter=torch.utils.data.DataLoader(mnist_test,batch_size=batch_size,shuffle=True,num_workers=num_
workers)
```

（2）查看数据集的数据维度

选中数据集中第一个数据，查看其数据维度。

参考代码:

```
feature, label = mnist_train[0]#读取第一个获得的训练数据，每一个训练数据均为 data 与 label
print(feature.shape, label)  #每一个 data 为三通道数据,label 为单通道标签数据
```

数据输出结果如下:

```
torch.Size([1, 28, 28]) 9
```

4.定义激活函数

这里利用基础的 max 函数来实现 ReLU，也可直接调用 PyTorch 中的 ReLU 函数。

参考代码:

```
def relu(X):#定义了激活函数
    return torch.max(input=X, other=torch.tensor(0.0))
```

5.定义 softmax 函数

定义 softmax 函数，将预测数据转换为预测概率分布，方便计算真实值和预测值之间的误差，将在模型训练后，输出任一样本特征类别的概率。

参考代码:

```
def softmax(X):#通过 softmax 函数，计算概率
    X_exp = X.exp()
    partition = X_exp.sum(dim=1, keepdim=True)
    return X_exp / partition  # 这里应用了广播机制
```

6.建立神经网络模型

Fashion-MNIST 数据集中图像为 28×28 像素，类别数为 10，但输入进网络前需要将数据转换成长度为 28×28=784 一维向量，隐藏层单元个数为 256。

参考代码:

```
num_inputs, num_outputs, num_hiddens = 784, 10, 256
W1 = torch.tensor(np.random.normal(0, 0.01, (num_inputs, num_hiddens)), dtype=torch.float)#采用初始化参数
b1 = torch.zeros(num_hiddens, dtype=torch.float)#采用零值初始化偏差
W2 = torch.tensor(np.random.normal(0, 0.01, (num_hiddens, num_outputs)), dtype=torch.float)
b2 = torch.zeros(num_outputs, dtype=torch.float)#采用零值初始化偏差

params = [W1, b1, W2, b2]#参数所构成的列表
for param in params:
    param.requires_grad_(requires_grad=True)#设置所有的参数列表中的参数均需要进行反向求导


def net(X):#定义网络结构
    X = X.view((-1, num_inputs))#将输入的图像数据展平为单列的 784 的数据，有利于全连接层的使用
    H = relu(torch.matmul(X, W1) + b1)#将 H ＝X*W1+b1，再求激活函数
    O = softmax(torch.matmul(H, W2) + b2)#将 O ＝H*W 2 ＋b2，再用 softmax 进行分类
    return O
```

7.定义损失函数

其中变量 y_hat 是样本在类别的预测概率，变量 y 是该样本的标签类别。通过 gather 函数，得到样本的标签的预测概率。

参考代码:

```
def cross_entropy(y_hat, y):#定义交叉熵损失函数
    return - torch.log(y_hat.gather(1, y.view(-1, 1)))
    #tensor.gather 与 torch.gather 用法一样，
    # tensor.gather(dim,indexs)，在 dim 维度上，安装 indexs 所给的坐标选择元素，返回一个和 index 维度相同的 tensor
```

8.定义优化函数

采用随机梯度下降算法不断迭代使得模型参数不断优化，使预测值逐渐接近真实值。

参考代码:

```
def sgd(params, lr, batch_size):#参数优化方法
```

```
    for param in params:#所有需要优化的参数
        param.data -= lr * param.grad / batch_size #  由于 param 是 tensor 格式，因此更改 param 参数数据时用的 param.data
```

9.定义精度评价函数

参考代码:

```
def evaluate_accuracy(data_iter, net):#精度评价
    acc_sum, n = 0.0, 0
    for X, y in data_iter:
        acc_sum += (net(X).argmax(dim=1) == y).float().sum().item()#dim=1 表示按行，取概率最大的，由于一次 X 包括了 batch 个数据，
        # 因此，把数据求和．.item()取出单元素张量的元素值并返回该值，保持原元素类型不变
        n += y.shape[0]
    return acc_sum / n
```

10.定义模型训练函数

采用小批量随机梯度下降法来优化模型的损失函数。在训练模型时，迭代周期数 num_epochs 和学习率 lr 都是可以调节的超参数，改变它们的值可能会得到分类更准确的模型。

参考代码:

```
num_epochs, lr = 5, 0.1#定义训练的次数与学习率

def train_ch2(net, train_iter, test_iter, loss, num_epochs, batch_size,
                  params=None, lr=None, optimizer=None):
    for epoch in range(num_epochs):
        train_l_sum, train_acc_sum, n = 0.0, 0.0, 0#每次训练对上次训练的结果进行归零
        for X, y in train_iter:
            y_hat = net(X)#
            l = loss(y_hat, y).sum()#l 要求偏导，是 tensor 格式

            # 梯度清零
            if optimizer is not None:#如果定义了特定的参数优化器，则将参数优化器中的参数设置为 0 ．本部分在使用的时候由于没有定义，所以不会执行下面的清零
                optimizer.zero_grad()
            elif params is not None and params[0].grad is not None:#如果参数有定义并且参数第一个值的梯度也有定义，则执行下面的清零
                for param in params:
                    param.grad.data.zero_()

            l.backward()#梯度反向求导
            if optimizer is None:
```

```
                sgd(params, lr, batch_size)#没有定义特殊的优化器，则使用默认的 SGD,本实验采用的是 sgd
            else:
                optimizer.step()#定义了特殊的优化器，则直接用优化器 step 进行参数的学习


            train_l_sum += l.item()#把所有损失函数的值相加，即 batch=256 个值相加起来
            train_acc_sum += (y_hat.argmax(dim=1) == y).sum().item()#精度是由预测出的概率最大项与标签之间对比相同，
            n += y.shape[0]#即为 batch 值
        test_acc = evaluate_accuracy(test_iter, net)#验证数据集的精度，求解的方法与训练的数据集方法一样
        print('epoch %d, loss %.4f, train acc %.3f, test acc %.3f'
              % (epoch + 1, train_l_sum / n, train_acc_sum / n, test_acc))#输出训练的次数，平均损失函数的大小，训练精度，测试精度
```

11.训练模型

调用模型训练函数训练模型。

※备注：模型训练中，需等待几分钟。

参考代码:

```
train_ch2(net, train_iter, test_iter, cross_entropy, num_epochs, batch_size, params, lr)
```

通过 5 个 epoch 后的模型训练结果如下:

```
epoch 1, loss 1.0478, train acc 0.636, test acc 0.754
epoch 2, loss 0.6055, train acc 0.787, test acc 0.808
epoch 3, loss 0.5204, train acc 0.818, test acc 0.801
epoch 4, loss 0.4839, train acc 0.830, test acc 0.830
epoch 5, loss 0.4602, train acc 0.839, test acc 0.834
```

从上面的训练结果可知该模型在训练集和测试集上的准确率越来越高，且 loss 值逐步递减。

五、实验结论

通过本实验可以了解到神经网络由输入层，隐藏层，输出层，网络越复杂，隐藏层的层数就越多。为了解决神经网络的非线性问题，通常需要增加激活函数。模型的输出结果是通过损失函数进行衡量，而梯度下降法是最常用的损失函数优化的方法。通过本次实验能够掌握神经网络的基本构建过程，以及深刻理解神经网络的原理。

3　基于神经网络正则化的服装分类

一、实验目的

（1）掌握神经网络的激活函数、损失函数、输出层、最优化；
（2）掌握多层神经网络的结构、原理；
（3）了解神经网络中过拟合问题解决技术。
（4）了解 PyTorch 下相关概念与函数；
（5）掌握 Dropout 正则化技术。

二、实验内容

神经网络训练需要通过大量的数据对模型进行学习，在此过程中，模型容易受到过拟合、欠拟合问题限制，因此，针对上述问题，通过模型结构的优化，数据的预处理、以及其他方法可有效缓解上述问题。

本实验选择 pytorch 系统中自带的服装分类数据集，通过对过拟合和欠拟合问题进行复现，并通过网络正则化优化，最终比较分类效果。

本实验针对神经网络模型过拟合问题，通过 Pytorch 建立神经三层网络模型，利用服装数据集 Fashion-Mnist 数据进行训练，采用梯度下降法和 Dropout 正则化技术对模型进行优化；最后对模型进行训练，获取模型在训练集和测试集上的准确率和损失函数值。

三、实验原理

模型训练的过程中经常面临两个典型的问题，一个是神经网络模型在训练的过程中无法得到较低的误差值，即欠拟合；另一个则是神经网络模型的训练误差远小于测试误差，即过拟合。产生这两种问题的原因有很多，本实验以多层感知机为例讨论如何防止出现过拟合的情况。

1. 欠拟合

欠拟合一般发生在模型训练刚开始的时候，一般随着模型迭代训练，欠拟合就不再考虑。实际上欠拟合是指模型在拟合数据集的过程中不能得到较低的误差，也就是说，模型复杂度低，训练的模型在训练集上表现很差，完全没有拟合到训练数据集的规律。

解决欠拟合的方法如下：

（1）模型复杂化：对同一个算法复杂化，例如回归模型添加更多的高次项，增加决策树的深度，增加神经网络的隐藏层数和隐藏单元数等； 弃用原来的算法，使用一个更加复杂的算法或模型，例如用神经网络来替代线性回归，用随机森林来代替决策树

等

（2）增加更多的特征，使输入数据具有更强的表达能力：特征挖掘十分重要，尤其是具有强表达能力的特征，往往可以抵过大量的弱表达能力的特征。特征的数量往往并非重点，质量才是，总之强特最重要。能否挖掘出强特，还在于对数据本身以及具体应用场景的深刻理解，往往依赖于经验。

（3）调整参数和超参数：神经网络中超参数包括学习率、学习衰减率、隐藏层数、隐藏层的单元数、Adam 优化算法中的β1 和β2 参数、batch_size 数值等。

其它算法中超参数包括随机森林的树数量、k-means 中的 cluster 数、正则化参数λ等。

2. 过拟合

过拟合指训练误差和测试误差太大，也就是说模型复杂度高，模型在训练的过程中出现在训练数据集上表现优良，在测试数据集上表现不足的情况，出现过拟合的情况主要如下:

（1）训练数据集样本单一，样本丰富度不够；

（2）训练数据集噪声干扰大；

（3）训练模型复杂度过高。

要解决过拟合的情况，提高过模型的泛化能力，除了获取更多的数据集、选择合适的模型以外，还可以采用正则化技术：L1 正则化、L2 正则化、Dropout。

L1 正则化:

通常又被称为 Lasso 回归，与一般的线性回归的区别在于 L1 正则化在损失函数增加了一个 L1 正则化的项，L1 正则化的项有一个常系数λ来调节损失函数的均方差项和正则化项的权重。

$$J(\theta)=\frac{1}{2}\sum_{i=0}^{k}(f_\theta(x^i)-y^i)^2+\lambda\sum_{j=1}^{m}\left|\theta_j\right|$$

通过 L1 正则化可以使一部分权重为零，因此产生一部分失活特征值，达到降低模型复杂度的目的。

L2 正则化:

通常又被称为 Ridge 回归，与一般的线性回归的区别在于 L2 正则化在损失函数增加了一个 L2 正则化的项，与 Lasso 回归相比，Ridge 回归的正则化项是 L2 范数，具体的 L2 正则化公式是:

$$J(\theta)=\frac{1}{2}\sum_{i=0}^{k}(f_\theta(x^i)-y^i)^2+\frac{\lambda}{2}\sum_{j=1}^{m}\theta_j^2$$

Ridge 回归在不抛弃任何一个特征的情况下，缩小了回归系数，使得模型相对而言

比较稳定，但和 Lasso 回归相比，这会使得模型特征留得特别多，模型解释性差。

Dropout

Dropout 是在训练网络时用的一种技巧（trick），相当于在隐藏单元增加了噪声。Dropout 指的是在训练过程中每次按一定的概率（比如 50%）随机地“删除”一部分隐藏单元（神经元）。所谓的“删除”不是真正意义上的删除，其实就是将该部分神经元的激活函数设为 0（激活函数的输出为 0），让这些神经元不计算而已。Dropout 相当于一个小型的模型融合。

那么到底是怎么实现的？简单来说就是在前向传导的时候，让某个神经元的激活值以一定的概率 p，让其逐渐停止工作，Dropout 的出现可以很好地解决这个问题，每次做完 dropout，相当于从原始的网络中找到一个更瘦的网络，示意图如下

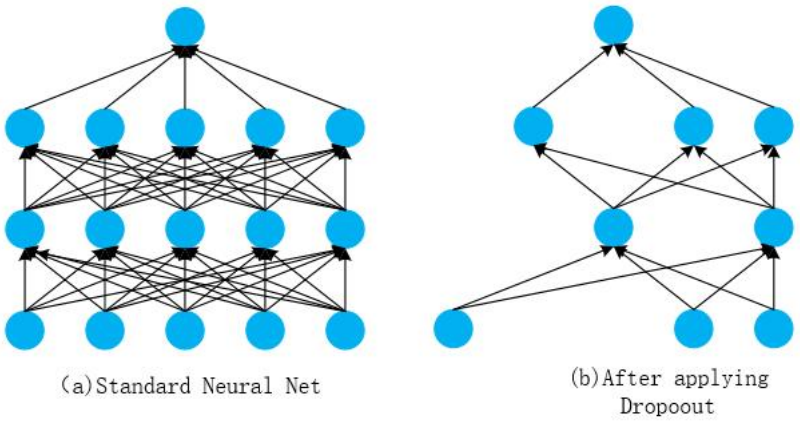


图 1　dropout 作用示意图

四、实验步骤

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

2.导入库文件及相关函数

引入 Torchvision 相关库，利用 Torchvision 相关模块的对象与函数完成数据集的建立；引入 numpy 库用于科学计算；引入 matplotlib.pyplot 库用于绘图。

参考代码:

```
%matplotlib inline
import torch
import torchvision
import torch.nn as nn
import numpy as np
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
```

3.建立数据集

(1)下载 Fashion-Mnist 服装数据集

※备注：下载数据集，需要等待几分钟。

通过 torchvision.datasets.FashionMNIST 函数下载 Fashion-Mnist 数据集，该数据集包括训练集和测试集两个数据集，包含 10 个类别，分别为：t-shirt（T 恤）、trouser（裤子）、pullover（套衫）、dress（连衣裙）、coat（外套）、sandal（凉鞋）、shirt（衬衫）、sneaker（运动鞋）、bag（包）和 ankle boot（短靴），并将其对应为相应的标签，且数据集为灰度图像，因此通道数为 1。

参考代码:

```
batch_size = 256
num_workers = 4
#下载训练集到指定目录
mnist_train = torchvision.datasets.FashionMNIST(root='/home/retoo/Desktop/实验/数据集/3.深度学习/3FashionMNIST', train=True, download=True, transform=transforms.ToTensor())
# 下载测试集到指定目录
mnist_test = torchvision.datasets.FashionMNIST(root='/home/retoo/Desktop/实验/数据集/3.深度学习/3FashionMNIST', train=False, download=True, transform=transforms.ToTensor())
# 将训练集组成一个批次
train_iter = torch.utils.data.DataLoader(mnist_train, batch_size=batch_size, shuffle=True,
num_workers=num_workers)
# 将测试集组成一个批次
test_iter = torch.utils.data.DataLoader(mnist_test, batch_size=batch_size, shuffle=True,
num_workers=num_workers)
```

（2）查看数据集的数据维度

选中数据集中第一个数据，查看其数据维度。

参考代码:

```
feature, label = mnist_train[0]#读取第一个获得的训练数据，每一个训练数据均为 data 与 label
print(feature.shape, label)  # Channel x Height x Width   # 每一个 data 为三通道数据,label 为单通道标签数据
```

输出结果如下:

```
torch.Size([1, 28, 28]) 9
```

4.定义 Dropout 函数

（1）定义 Dropout 函数

若将所有的神经元丢弃，则将 keep_prob 设置为 0，否则就以一定的概率对神经元进行丢弃。

参考代码:

```
def dropout(X, drop_prob):#定义 droppout 模块
    X = X.float()
    assert 0 <= drop_prob <= 1#设置丢失的概率，如果不是 0 ~1,则系统设置错误
    keep_prob = 1 - drop_prob
    # 这种情况下把全部元素都丢弃
    if keep_prob == 0:
        return torch.zeros_like(X)#如果丢失率为 1 0 0 %,则返回的值全部为零
    mask = (torch.rand(X.shape) < keep_prob).float()  # 丢失概率不为全部，X.shape([2,8]),随机生产一个 0 —1 的形状与 X 相同的，小于 0． 5 的为 1，否则为 0
    # 通过 m a s k *X，保留 1 位置的值/home/retoo/Desktop/实验/数据集/3.深度学习/3FashionMNIST', train=False, download=True, transform=transforms.ToTensor())

    # 为了维持训练和测试的时候输出的期望保持一致，在代码上有两种做法，一种是在训练的时候对输出除以 1-p（保留的概率），另外一种则是在预测的时候对输出乘以 p

    return mask * X / keep_prob
```

（2）利用 torch 初始化输入数据，验证 Dropout 丢弃效果

参考代码:

```
X = torch.arange(16).view(2, 8)#假设输入的图像为 1 6 个，将其 resize 到(2,8)
dropout(X, 0.5)
```

Dropout 结果如下:

```
tensor([[ 0.,  2.,  4.,  0.,  8.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.,  0.,  0., 30.]])
```

从以上输出结果可看出，当设置丢弃率为 0.5 时，有将近一半的特征值被置为零。

5.定义神经网络模型参数

Fashion-MNIST 数据集中图像为 28×28 像素，类别数为 10，但输入进网络前需要将数据转换成长度为 28×28=784 一维向量，隐藏层单元个数为 256。

参考代码:

```
num_inputs, num_outputs, num_hiddens1, num_hiddens2 = 784, 10, 256, 256#设置各层的网络节点数
```

```
W1 = torch.tensor(np.random.normal(0, 0.01, size=(num_inputs, num_hiddens1)), dtype=torch.float, requires_grad=True)
#W1 为第一层的参数，输入节点与第一层隐含层之间的参数
b1 = torch.zeros(num_hiddens1, requires_grad=True)#第一层的偏置量
W2 = torch.tensor(np.random.normal(0, 0.01, size=(num_hiddens1, num_hiddens2)), dtype=torch.float, requires_grad=True)
#W 2 为第二层的参数，第一层隐含层与第二层隐含层之间的参数
b2 = torch.zeros(num_hiddens2, requires_grad=True)#第二层的偏置量
W3 = torch.tensor(np.random.normal(0, 0.01, size=(num_hiddens2, num_outputs)), dtype=torch.float, requires_grad=True)
#W 3 为第三层的参数，第二层隐含层与输出层之间的参数
b3 = torch.zeros(num_outputs, requires_grad=True)#第三层的偏置量
params = [W1, b1, W2, b2, W3, b3]#参数列表，但是每个参数均为 tensor
```

6.定义神经网络模型

定义含两层隐藏层的多层感知机，这里将全连接层和激活函数连起来使用，并对每个激活函数后的特征值使用丢弃法，分别设置不同层的丢弃率，一般情况下，在开始层时采用较低的丢弃率，在本实验中，分别将第一个隐藏层的丢弃率设置为 0.2，将第二个隐藏层的丢弃率设置为 0.5，由于训练模式和测试模式下需要判断是否开启 Dropout，一般测试模式时默认关闭。

参考代码:

```
drop_prob1, drop_prob2 = 0.2, 0.5#不同的 droppout 比例
def net(X, is_training=True):
    X = X.view(-1, num_inputs)#将输入展平为一维的
    H1 = (torch.matmul(X, W1) + b1).relu()
    if is_training:  # 只在训练模型时使用丢弃法
        H1 = dropout(H1, drop_prob1)  # 在第一层全连接后添加丢弃层
    H2 = (torch.matmul(H1, W2) + b2).relu()
    if is_training:
        H2 = dropout(H2, drop_prob2)  # 在第二层全连接后添加丢弃层
    return torch.matmul(H2, W3) + b3
```

7.定义损失函数

损失函数直接调用 torch.nn 中的交叉熵损失函数。

参考代码:

```
loss = torch.nn.CrossEntropyLoss()#损失函数直接选择 torch.nn 中的交叉熵损失函数
```

8.定义优化函数

采用随机梯度下降算法不断迭代使得模型参数不断优化，使预测值逐渐接近真实值。

参考代码:

```
def sgd(params, lr, batch_size):
    for param in params:
        param.data -= lr * param.grad / batch_size # 注意这里更改 param 时用的 param.data
```

9.定义精度评价函数

在训练的过程中需要同时统计测试集的预测情况来监控模型是否出现过拟合或者欠拟合的情况。

参考代码:

```
def evaluate_accuracy(data_iter, net):
    acc_sum, n = 0.0, 0
    for X, y in data_iter:
        if isinstance(net, torch.nn.Module):#如果 net 模型为基于 torch.nn.Module 搭建的，深度学习的常用模型
            net.eval() # 评估模式，这会关闭 dropout
            acc_sum += (net(X).argmax(dim=1) == y).float().sum().item()
            net.train() # 改回训练模式
        else: # 自定义的模型
            if('is_training' in net.__code__.co_varnames): # 如果有 is_training 这个参数
                #net.__code__.co_varnames 将函数的局部变量以元组的形式返回，除此以外，还有 fun.__code__.co_argcount 返回函数中参数的个数等（*args 以前的）
                # 将 is_training 设置成 False
                #?为什么在精度计算的过程中需要将网络修改为验证模式
                acc_sum += (net(X, is_training=False).argmax(dim=1) == y).float().sum().item()
            else:
                acc_sum += (net(X).argmax(dim=1) == y).float().sum().item()
        n += y.shape[0]
    return acc_sum / n
```

10.定义模型训练函数

采用小批量随机梯度下降法来优化模型的损失函数。在训练模型时，迭代周期数 num_epochs 和学习率 lr 都是可以调节的超参数，改变它们的值可能会得到分类更准确的模型。

参考代码:

```
num_epochs, lr = 5, 0.1

def train_ch3(net, train_iter, test_iter, loss, num_epochs, batch_size,
              params=None, lr=None, optimizer=None):
    for epoch in range(num_epochs):
        train_l_sum, train_acc_sum, n = 0.0, 0.0, 0#训练的时候，每个 epoch 都需要对其进行清零
```

```
        for X, y in train_iter:
            y_hat = net(X)
            l = loss(y_hat, y).sum()#l 还是为 tensor，值是一个.

            # 梯度清零
            if optimizer is not None:#如果定义了优化器
                optimizer.zero_grad()
            elif params is not None and params[0].grad is not None:#如果定义了参数
                for param in params:
                    param.grad.data.zero_()
            l.backward()#梯度反向求导
            if optimizer is None:
                sgd(params, lr, batch_size)#没有定义优化器，用自己定义的 sgd 求解参数
            else:
                optimizer.step()  #优化器为 torch 中提供的，则用.step()
            train_l_sum += l.item()#每次生成的 batch 个损失函数，在一次 epoch 中集中计算一次
            train_acc_sum += (y_hat.argmax(dim=1) == y).sum().item()#在每次的预测输出中，通过
概率最大值与标签值之间的比较
            n += y.shape[0]#batch 值
        test_acc = evaluate_accuracy(test_iter, net)#验证集数据的精度计算
        print('epoch %d, loss %.4f, train acc %.3f, test acc %.3f'
              % (epoch + 1, train_l_sum / n, train_acc_sum / n, test_acc))
```

11.训练模型

调用模型训练函数训练模型。

※备注：模型训练中，需等待几分钟。

参考代码:

```
num_epochs, lr = 5, 100.0
train_ch3(net, train_iter, test_iter, loss, num_epochs, batch_size, params, lr)
#对比实验 2，本实验中损失函数更小，训练的精度也要更高些.
```

通过 5 个 epoch 后的模型训练结果展示如下:

```
epoch 1, loss 0.0046, train acc 0.547, test acc 0.754
epoch 2, loss 0.0023, train acc 0.784, test acc 0.807
epoch 3, loss 0.0020, train acc 0.820, test acc 0.786
epoch 4, loss 0.0018, train acc 0.837, test acc 0.769
epoch 5, loss 0.0016, train acc 0.849, test acc 0.838
```

由于数据读取的过程中具有随机性，因此实验结果具有一定的波动，建议多实验几次。

五、实验结论

本实验主要是介绍神经网络的正则化，而之所以要对神经网络进行正则化就是为了解决神经网络的过拟合。神经网络在训练的过程中会产生过拟合和欠拟合，而解决欠拟合一般是增加数据量和优化模型，而过拟合通俗来讲就是模型把数据学习得太彻底，以至于把噪声数据的特征也学习到了，这样就会导致在后期测试的时候不能够很好地识别数据，即不能正确的分类，要解决这个问题常用的方式有 L1 正则化、L2 正则化，Dropout 等，希望在实际应用中能够灵活运用这几种方法去防止过拟合。

图像的代数运算

一、实验目的

（1）掌握代数运算的基本原理及用途；

（2）熟悉 cv2 中的代数运算模块的使用与操作；

（3）熟悉 Python 中 Matpiotilb 库中绘图库的使用和操作；

（4）掌握 Matpiotilb 图形的可视化。

二、实验内容

图像代数运算是指对两幅或两幅以上的输入图像进行加减乘除四则运算。它在图像处理中有着广泛的应用，加法运算可以用来降低图像中的随机噪声，减法运算可以用来减去背景、运动背景、进行梯度幅度，乘法运算通常用来进行掩模，除法运算可以用来归一化。

本实验要求将两幅图像进行加法、减法、乘法、除法运算，显示运算结果。

三、实验原理

在数字图像处理过程中，为了达到理想图像处理效果或需对图像某些特征进行突出，有时需要对图像进行运算及变换。代数运算是指两幅图像对应像素的加减乘除运算，这些基本的代数运算可以完成组合使用，组合而成的运算为复合代数运算。在处理图像是，代数运算可以抑制或消除噪声，也可以利用叠加运算合成新的图像。

图像处理代数运算的四种基本形式分别如下：

$$f(x,y)=h(x,y)+g(x,y)$$

$$f(x,y)=h(x,y)-g(x,y)$$

$$f(x,y)=h(x,y)*g(x,y)$$

$$f(x,y)=h(x,y)/g(x,y)$$

其中 $h(x,y)$ 和 $g(x,y)$ 分别为两幅输入图像在 (x,y) 处的的灰度值或彩色值。这些运算，都是对应像素运算，即这些运算是在 h 和 g 对应的像素对之间执行的，其中 x=0，1，2，…,M-1， y=0，1，2，…，N-1，通常，M 和 N 分别是图像对应的行数和列数，代数运算即像素位置不变，将对应像素的灰度值或彩色份量进行计算。

1. 加法运算

加运算就是将两幅图像对应像素的灰度值或彩色分量进行相加。图像相加经常有两

种用途，即消除图像的随机噪声，主要方法是将同一场景的图像相加后再取平均；另外一种是做特效，把多幅图叠加在一起再进行进一步处理。

假设 g(x,y)是无噪声图像 f(x,y)被加性噪声η(x,y)污染后的图像，即

$$g(x,y)= f(x,y)+ \eta(x,y)$$

其中假设在每对坐标(x,y)处，噪声是不相关的，并且均值为零，还假设噪声和图像值也是不相关的（对于加性噪声，这是典型的假设），可以通过一组带噪输入图像$\{g_i(x,y)\}$相加来降低输出图像的噪声含量，这也是图像增强频繁使用的技术。

如果噪声满足刚才声明的约束条件,若图像 $\bar{g}(x,y)$ 是通过对 k 幅不同的噪声图像取平均得到的：

$$\bar{g}(x,y)=\frac{1}{K}\sum_{i=1}^{k} g_i(x,y) \tag{1}$$

则其满足

$$E\{\bar{g}(x,y)\}=f(x,y) \tag{2}$$

和

$$\sigma^2_{\bar{g}(x,y)}=\frac{1}{K}\sigma^2_{\eta(X,Y)} \qquad \eta(x,y)$$

(3)

式中，是 $\bar{g}(x,y)$ 的期望值，$\sigma^2_{\bar{g}(x,y)}$和$\sigma^2_{\eta(x,y)}$分别是 $\bar{g}(x,y)$ 和$\eta(x,y)$ 在坐标(x,y)处的方差。这些方差是与输入图像大小相同的阵列，并且每个像素位置有一个标量方差。

平均图像中任意一点(x,y)处的标准差是

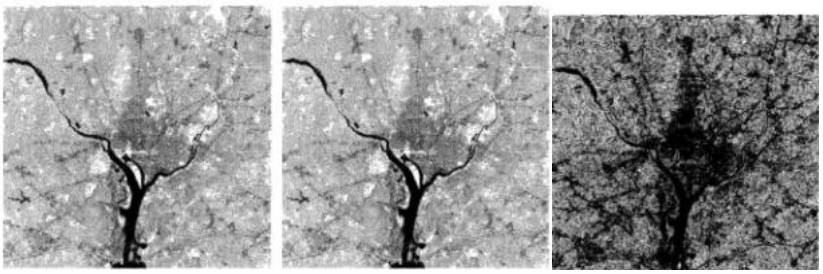
$$\sigma_{\bar{g}(x,y)}=\frac{1}{\sqrt{K}}\sigma_{\eta(x,y)} \tag{4}$$

K 增大时,式(3)和式(4)表明每个位置(x,y)的像素值的变化性(由方差或标准差度量)将减小。因为$E\{\bar{g}(x,y)\}=f(x,y)$，这意味着平均处理中所用的噪声图像的数量增加时，$\bar{g}(x,y)$ 将逼近 f(x,y)。为避免输出(平均)图像出现模糊和其他人为失真，需要对图像 $g_i(x,y)$ 进行配准(即空间上对齐)

对于灰度图像，相加结果为对应像素的灰度值相加，如果是彩色图像，则为对应颜色的分量相加。需要注意的是，像素值的区间在[0，255]如果两个数使用运算符号直接相加之和大于 255，那么将运算结果取模如：255+58=313，则相加后的像素值（255+58）%256=57，得到计算结果 57，如果使用 opencv 中的库函数 cv2.add(),如果两数相加大于 255，那么运算结果将取饱和值 255。

2. 减法运算

图像减法运算将两幅图像对应像素的灰度值或彩色分量进行相减，可以用于运动目标检测，还可以通过图像相减来比较图像，例如，图 (b)中的图像是通过把图 (a)中的每个像素的最低有效位设置为 0 得到的。这两幅图像视觉上很难区分。然而，如图(c)所示，从一幅图像中减去另--幅图像，可清楚地显示了它们的差。在差值图像中，黑(0)值指出了图(a)和图(b)之间没有差别的位置。



图(a)原图　　图(b)预处理后的图　　图(c)差值图

图 1 减法运算效果图

3. 乘法运算

图像乘法运算就是将两幅图像对应像素的灰度值或彩色分量进行相乘。乘运算的主要作用就是抑制图像的某些区域，对于需要保留下来的区域，掩模置为 1，否则置为 0。乘运算有时候也可以用来实现卷积或相关运算。

4. 除法运算

图像除法运算时将两幅图像对应像素的灰度值或彩色分量值进行相除。简单的除法运算可用于改变图像的灰度级。

对于乘法和除法运算图像最重要的应用是阴影校正。假设成像传感器产生的图像可建模为由 $f(x,y)$表示的“完美图像”与阴影函数 $h(x,y)$的乘积，即 $g(x,y)=f(x,y)h(x,y)$。若 $h(x,y)$已知或可以估计，则可用 $h(x,y)$的反函数[即采用对应像素除法的 g 除以 h]乘以感测图像的方法得到 $f(x,y)$(或它的一个估计)。若能访问成像系统，则可通过对恒定灰度目标成像，得到阴影函数的一个良好近似。

四、实验步骤

1. 运行 jupyter lab

（1）在“实验”文件夹空白处单击鼠标右键，选择“打开终端”，在终端界面输入“jupyter lab”。

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器。

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为所做实验名。

2.图像加法运算

cv2.add 函数用于两幅图像进行相加。cv2.addWeighted 函数用于两张相同分辨率、相同文件类型的图片融合。

参考代码：

```
# 导入库文件及函数
import cv2
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt

#  图像加法
p1 = cv2.imread("/home/retoo/Desktop/实验/数据集/4.数字图像处理/1-1.jpg")
p2 = cv2.imread("/home/retoo/Desktop/实验/数据集/4.数字图像处理/1-2.jpg")
res1 = cv2.add(p1, p2)
res1_1 = p1 + p2
# 图像融合
res2 = cv2.addWeighted(p1, 0.3, p2, 0.7, 0)
# 显示原图像及不同加法处理后的图像
plt.figure(figsize=(128, 32))
plt.subplot(2, 3,  1)
plt.imshow(cv2.cvtColor(p1, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(2, 3,  2)
plt.imshow(cv2.cvtColor(p2, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(2, 3,  3)
plt.imshow(cv2.cvtColor(res1, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(2, 3,  4)
plt.imshow(cv2.cvtColor(res1_1, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(2, 3,  5)
plt.imshow(cv2.cvtColor(res2, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

运行结果如下图 4 所示：

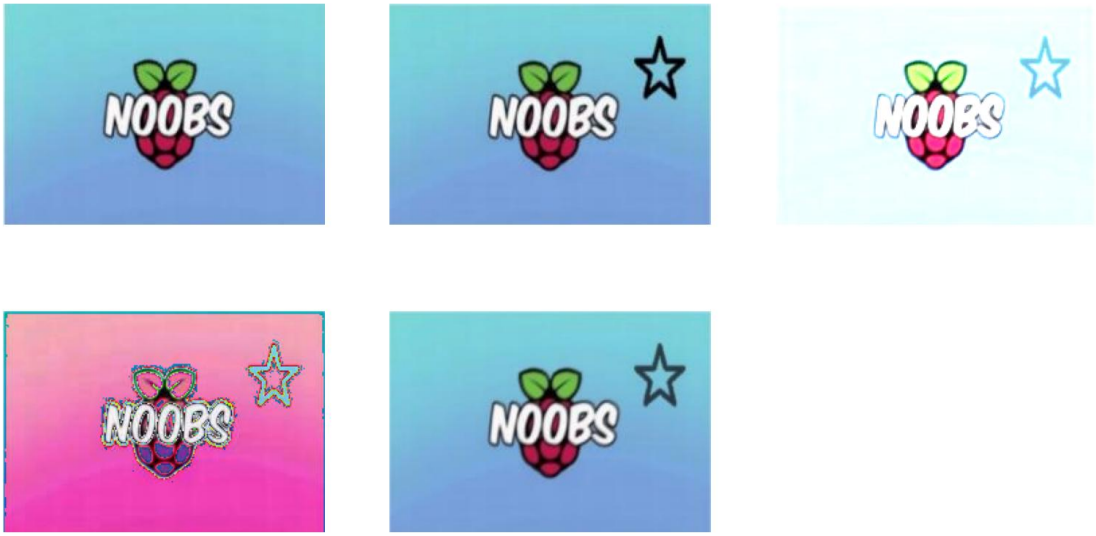


图 4　加法运算运行结果图

可以观察到直接使用运算符号进行图像相加的rest1_1的图像亮度显然是低于使用cv2.add()函数的rest1，这是由于使用运算符号相加使原来的值对256取模，会使其像素值可能会低于原值，降低了亮度，而使用cv2.add()函数通过取饱和值使像素值大于原像素值，反而提高了亮度。

4. 图像减法运算

cv2.subtract(src1,src2)函数与 cv2.absdiff(src1,src2)函数都用于计算 src1 与 src2 两幅图像之间的差异图像，两幅图像对像素的灰度值或彩色分量相减，用于目标检测。

cv2.absdiff()与 cv2.subtract()的区别是 cv2.absdiff()计算两幅图像差的绝对值。

参考代码：

```
# 导入库文件及函数
import cv2
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt

#   图像减法
p1 = cv2.imread("/home/retoo/Desktop/实验/数据集/4.数字图像处理/1-1.jpg")
p2 = cv2.imread("/home/retoo/Desktop/实验/数据集/4.数字图像处理/1-2.jpg")
# p1 与 p2 的差值
res1_1 = cv2.subtract(p1, p2)
# p2 与 p1 的差值
res1_2 = cv2.subtract(p2, p1)
# p1 与 p2 的差的绝对值(|p1-p2|=|p2-p1|)
res1_3 = cv2.absdiff(p1, p2)

res2 = cv2.addWeighted(p1, 1, p2, -1, 0)
```

```
# 显示原图像和不同减法处理后的图像
plt.figure(figsize=(128, 32))

plt.subplot(2, 3,  1)
plt.imshow(cv2.cvtColor(p1, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(2, 3,  2)
plt.imshow(cv2.cvtColor(p2, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(2, 3,  3)
plt.imshow(cv2.cvtColor(res1_1, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(2, 3,  4)
plt.imshow(cv2.cvtColor(res1_2, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(2, 3,  5)
plt.imshow(cv2.cvtColor(res1_3, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(2, 3,  6)
plt.imshow(cv2.cvtColor(res2, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

运行结果如下所示：

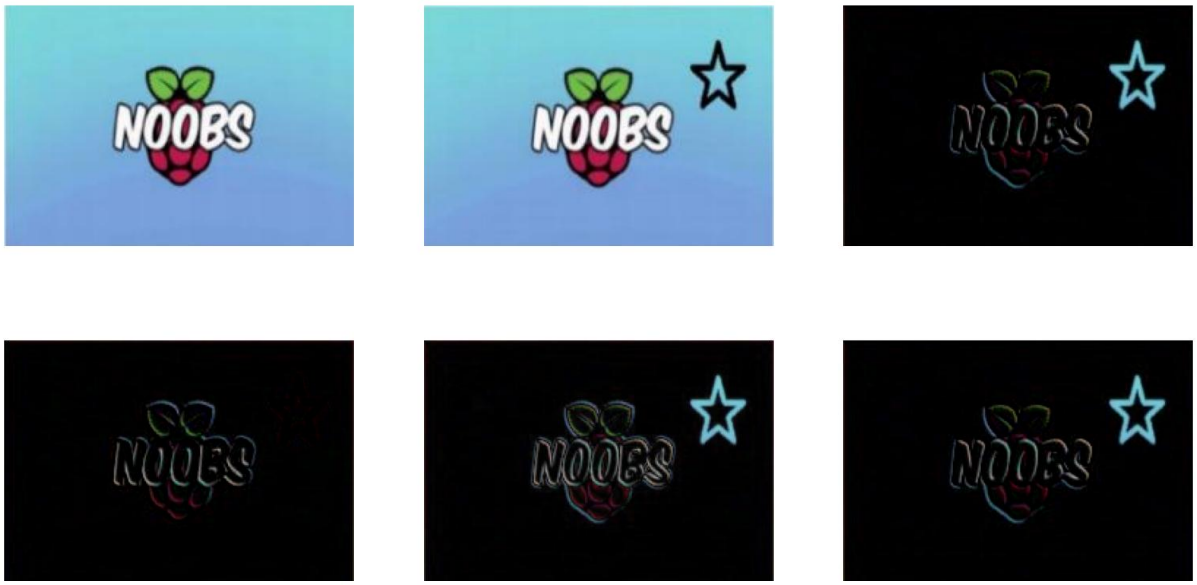


图 5 图像减法差值效果图

由于 cv2.subtract()函数对于所得值小于 0 则按 0 处理，所以 res1_2 的图像为黑色图像，这是因为像素值都为 0，而如果采用 cv2. absdiff 函数，由于所得结果为绝对值，将原本结果所得的负值变成了正值，所以所的图像将不会显示为黑色图像。

5. 图像乘法运算

cv2.multiply(src1, src2)函数用于将两幅图像对应的灰度值或彩色分量进行相乘。

参考代码：

```
# 导入库文件及函数
import cv2
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt

# 图像乘法运算
p1 = cv2.imread("/home/retoo/Desktop/实验/数据集/4.数字图像处理/1-1.jpg")
p2 = cv2.imread("/home/retoo/Desktop/实验/数据集/4.数字图像处理/1-2.jpg")
res = cv2.multiply(p1, p2)

# 显示原图像和乘法处理后的图像
plt.figure(figsize=(128, 32))
plt.subplot(1, 3,  1)
plt.imshow(cv2.cvtColor(p1, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(1, 3,  2)
plt.imshow(cv2.cvtColor(p2, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(1, 3,  3)
plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

程序运行结果如下所示：

图 6 图像乘法效果图

6. 图像除法运算

cv2.divide(src1, src2)函数用于将两幅图像对应的灰度值或彩色分量进行相除。

参考代码：

```
# 导入库文件及函数
import cv2
get_ipython().run_line_magic('matplotlib', 'inline')
```

```
import matplotlib.pyplot as plt

# 图像除法运算
p1 = cv2.imread("/home/retoo/Desktop/实验/数据集/4.数字图像处理/1-1.jpg")
p2 = cv2.imread("/home/retoo/Desktop/实验/数据集/4.数字图像处理/1-2.jpg")
res = cv2.divide(p1, p2)

# 通过原图像和除法处理后的图像
plt.figure(figsize=(128, 32))
plt.subplot(1, 3,  1)
plt.imshow(cv2.cvtColor(p1, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(1, 3,  2)
plt.imshow(cv2.cvtColor(p2, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(1, 3,  3)
plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

程序运行结果如下所示：

图 7 图像除法效果图

图像相乘和相除和图像的加法一样，也遵循“饱和运算”法则，对于大于 255 的值作为 255 处理，阴影矫正是图像的相乘和相除的重要应用。

五、实验结果分析

本实验通过加减乘除代数运算对图像进行处理，从实验结果可以看出在加法运算中直接代数相加和采用 cv2.add()函数进行图像相加时处理大于 255 的值有区别，乘除运算也遵循“饱和运算”法则，并了解到阴影矫正是乘除运算的重要应用。

图像操作之打码与解码

一、实验目的

（1）掌握 opencv 中的按位逻辑运算；

（3）掌握掩模运算原理及操作；

（3）掌握按位异或运算进行图像的加密和解密的原理；

（4）掌握图像打码与解码的原理。

二、实验内容

一个图像可以用像素矩阵来表示，通过随机数生成一个新的像素矩阵，称之为密钥图像，将原图与密钥图像进行按位异或运算，对图像进行加密，生成一个加密图像，这是对图像的整体打码。通常通过掩膜提取图像中感兴趣区域（ROI），再对 ROI 进行加密，这是对图像的局部打码。将加密图像与原密钥图像按位异或运算，进行解密，可以得到原图，这个过程叫做解码。

编写 Python 程序，将 2lena 头像中脸部感兴趣区域（ROI）打码、解码，并显示原图像及打码、解码过程中出现的图像。

三、实验原理

1. 按位逻辑运算

按位逻辑运算分为：按位与运算，按位或运算，按位非运算，按位异或运算。

（1）按位与运算

在与运算中，当参与与运算的两个逻辑都是真时，结果为真。其逻辑关系可类比为串联电路，只有当两个开关都闭合时，灯才会亮。在 opencv 中，可以使用 cv2bitwise_and() 函数来实现按位与运算，其语法格式为：

dst = cv2.bitwise_and(src1, src2[, mask])

dst 表示与输入值具有同样大小的 array 输出值；

src1 表示第一个 array 或 scalar 类型的输入值；

src2 表示第二个 array 或 scalar 类型的输入值；

mask 表示可选操作掩码，8 位通道 array。

按位操作有以下特点：将任何数值 N 与数值 0 进行按位与操作，都会得到数值 0。将任何数值 N(这里仅考虑 8 位值)与数值 255(8 位二进制数时 1111 1111)进行按位与操作都会的得到数值本身。

（2）按位或运算

或运算的规则是，当参与或运算的两个逻辑值中有一个为真，结果为真。

按位或运算是指将数值转换为二进值后，在对应位置上进行或运算。下表展示了两

个数值进行按位或运算的示例：

表 1 按位或运算表

数值	十进制	二进制
数值 1	198	1100 0110
数值 2	219	1101 1011
按位或运算结果	223	1101 1111

在 opencv 中，可以使用 cv2.bitwise_or()函数来实现按位或运算。

2. 按位异或运算

按位异或运算也叫做半运算，其运算法则与不带进位的二进制加法类似，其英文名为“exclusiveOR”,因此其函数通常表示为 xor。

表 2 异或运算表

算子 1	算子 2	结果	规则
0	0	0	xor(0,0)=0
0	1	1	xor(0,1)=1
1	0	1	xor(1,0)=1
1	1	0	xor(1,1)=0

按位异或运算是指将数值转化为二进制值后，在对应位置上进行异或运算。如下表所示：

表 3 按位异或运算表

数值	十进制值	二进制值
数值 1	198	1100 0110
数值 2	219	1101 1011
按位或运算结果	29	0001 1101

通过按位逻辑运算，我们可以在保证图像色彩和透明度不会变化的前提，将我们所需要两张或多张图像进行合成。

2．掩模

OpenCV 中的很多函数都会指定一个掩模，也被称作掩码，例如：

计算结果=cv2.add(参数 1，参数 2，掩模)

当使用掩模参数时，操作只会在掩模值为非空的像素点上执行，并将其他像素点的值置为 0。例如，img1，img2,mask 和 img3 的原始值分别为：

$$img1=\begin{bmatrix}3 & 3 & 3 & 3\\ 3 & 3 & 3 & 3\\ 3 & 3 & 3 & 3\\ 3 & 3 & 3 & 3\end{bmatrix}$$

$$img2=\begin{bmatrix}5&5&5&5\\5&5&5&5\\5&5&5&5\\5&5&5&5\end{bmatrix}$$

$$mask=\begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&1&1\\0&0&1&1\end{bmatrix}$$

$$img3=\begin{bmatrix}66&66&66&66\\66&66&66&66\\66&66&66&66\\66&66&66&66\end{bmatrix}$$

经过 img3=cv2.add(img1,img2,mask=mask)运算后，得到 img3 为：

$$img4=\begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&8&8\\0&0&8&8\end{bmatrix}$$

在运算过程中，img3 计算的是在掩模 mask 控制下的“imgl+img2”结果。在计算时，掩码为 1 的部分对应“img1+img2”，其他部分的像素值均为“0”。

需要说明的是，在运算前，数组 img3 内就存在值，这仅是为了说明问题用的，实际上 img3 是根据函数 cv2.ad0 所生成的新数组，与原来的值并没有关系。

上例中是采用 cv2.add()函数进行掩模的情况，位运算中也含有掩模参数，将彩色图像与掩模进行计算时，由于按位与操作要求参与运算的数据有相同的通道，所以无法直接将彩色图像与单通道的掩模图像进行按位与操作。通过将掩模图像转换为 BGR 模式的彩色图像，让彩色图像与(彩色)掩模图像进行按位与操作，从而实现掩模运算。实际上，在函数中所使用的掩模参数可以是 8 位单通道图像。所以，可以将掩模图像作为按位与函数 cv2.bitwiseand(src1,src2[,mask]])中参数 mask 的值，完成掩模运算。

此时，待处理的彩色图像同时作为函数参数，cv2.bitwise_and(src1,src2[,mask]])的参数 src1 和参数 src2，使用掩模图像作为掩模参数，进行按位与运算，即可得到由掩模控制的彩色图像。可以通过掩模来提取感兴趣的区域，或对图像中某些区域作屏蔽,或用相似性变量或图像匹配方法检测和提取图像中与掩模相似的结构特征。

3．加密和解密

通过按位异或运算可以实现图像的加密和解密。通过对原始图像与密钥图像进行按位异或运算，可以实现加密;将加密后的图像与密钥图像再进行按位异或运算，就可以对图像进行解密。根据异或运算规则，假设：

xor(a,b)=c

则可以得到：

xor(c,b)=a

xor(c,a)=b

如果上述 a，b，c 具有如下关系：a 明文，原始数据，b:密钥。c：密文，通过 xor(a,b)实现，则可以对上述数据进行如下操作和理解。

加密过程：将明文 a 与密钥 b 进行按位异或运算，完成加密，得到密文 c。

解密过程：将密文 c 与密钥 b 进行按位异或运算，完成解密，得到明文 a。

四、实验步骤

1.运行 jupyter lab

（1）在“实验”文件夹空白处单击鼠标右键，选择“打开终端”，在终端界面输入“jupyter lab”。

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器。

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为所做实验名。

2.导入库文件及函数

首先引入 Python 自带的 cv2 相关库，利用 cv2 相关库中的对象与函数完成对图像的处理。

```
import cv2
import numpy as np
```

3. 读取原图像，产生脸部感兴趣部分（ROI）的掩模矩阵

参考代码:

```
lena = cv2.imread("/home/retoo/Desktop/实验/数据集/4.数字图像处理/2lena.jpg") #读取原图像
mask=np.zeros(lena.shape,dtype=np.uint8) #产生与 lena 图像大 小相同的 mask 矩阵，初值为 0
mask[220:400,250:350]=1 #生成脸部感兴趣部分（ROI）的掩模矩阵
key=np.random.randint(0,256,lena.shape,dtype=np.uint8) #产生密匙图像 key
```

4.将原图像加密

```
key=np.random.randint(0,256,lena.shape,dtype=np.uint8) #产生密匙图像 key
#将原图像加密
lenaXorKey=cv2.bitwise_xor(lena,key) #将 lena 原图像与 key 密匙图像按位异或运算，进行加密
```

5．将加密后的图像打码

```
#将加密后的图像打码
encryptFace=cv2.bitwise_and(lenaXorKey,mask*255) #将脸部感兴趣部分（ROI）打码
noFace1=cv2.bitwise_and(lena,(1-mask)*255) #获取不含脸部感兴趣部分（ROI）的图像
maskFace=encryptFace+noFace1 #合成带打码的图像
```

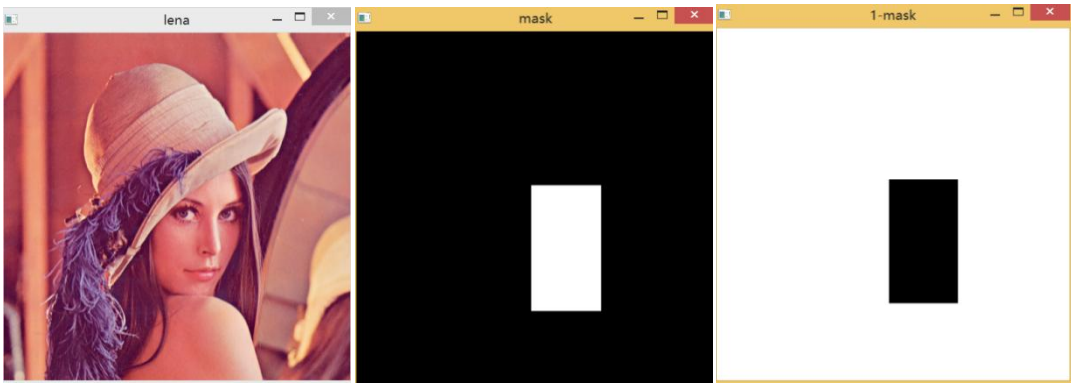
6.将加密打码后的图像解密、解码

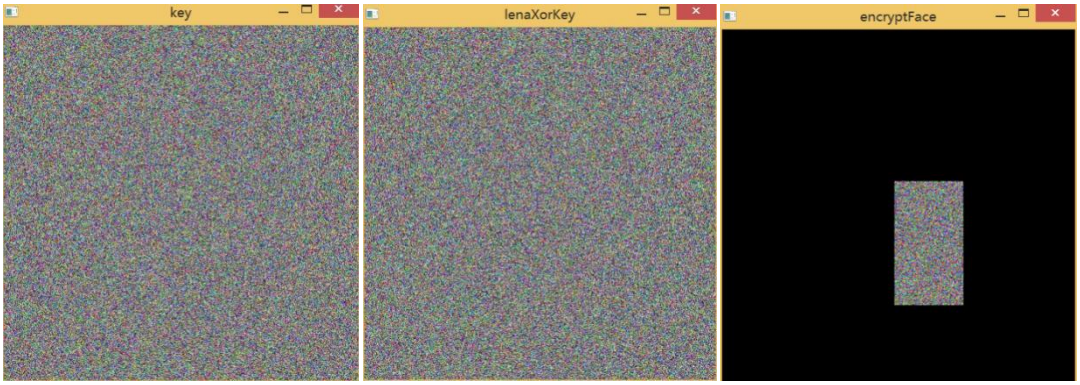
```
#将加密打码后的图像解码
extractOriginal=cv2.bitwise_xor(maskFace,key) #将加密打码后的图像 maskFace 与 key 密匙图像按位异或运算，进行解密
extractFace=cv2.bitwise_and(extractOriginal,mask*255) #将脸部感兴趣部分（ROI）解码
noFace2=cv2.bitwise_and(maskFace,(1-mask)*255) #获取不含脸部感兴趣部分（ROI）的解密图像
extractLena=noFace2+extractFace #合成解密解码后的图像
```

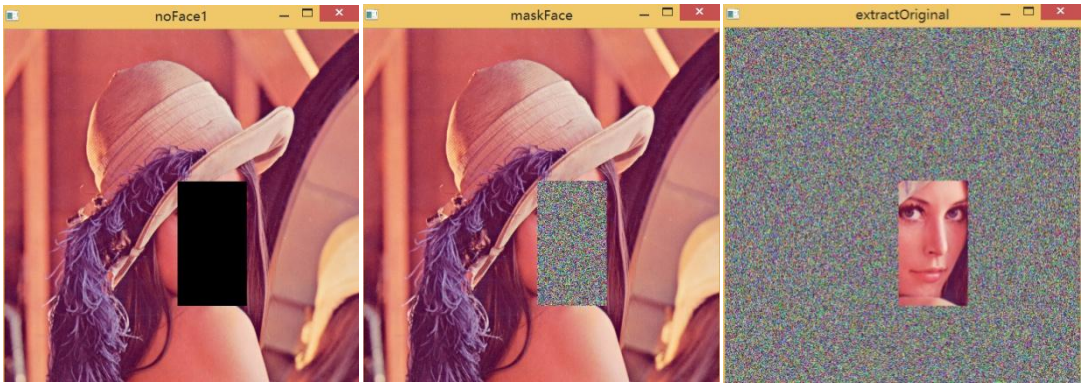
7. 显示图像

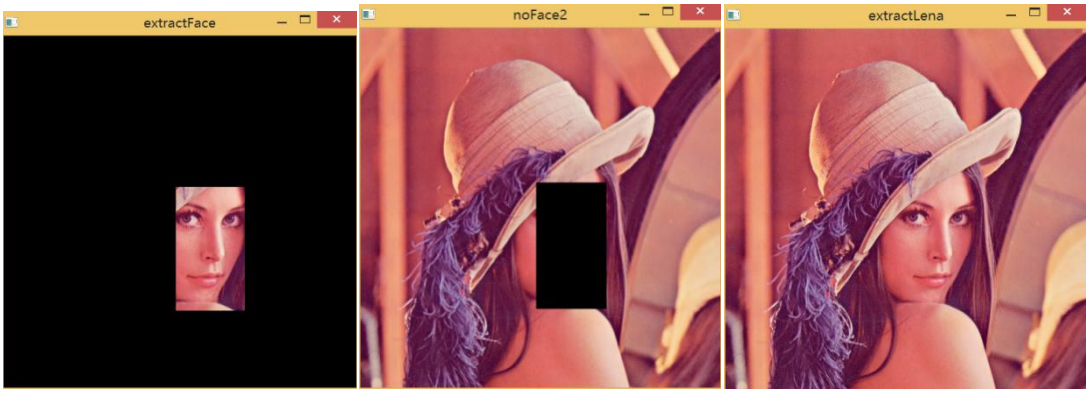
```
#显示图像
cv2.imshow("lena",lena) #显示原图像
cv2.imshow("mask",mask*255)#显示脸部感兴趣部分（ROI）的掩模矩阵
cv2.imshow("1-mask",(1-mask)*255)#显示不含脸部感兴趣部分（ROI）的矩阵
cv2.imshow("key",key)#显示密匙图像
cv2.imshow("lenaXorKey",lenaXorKey)#显示加密图像
cv2.imshow("encryptFace",encryptFace)#显示脸部感兴趣部分（ROI）打码图像
cv2.imshow("noFace1",noFace1)#不含脸部感兴趣部分（ROI）的图像
cv2.imshow("maskFace",maskFace)#显示带打码的图像
cv2.imshow("extractOriginal",extractOriginal)#显示解密图像
cv2.imshow("extractFace",extractFace)#显示脸部感兴趣部分（ROI）的解码图像
cv2.imshow("noFace2",noFace2)#显示不含脸部感兴趣部分（ROI）的解密图像
cv2.imshow("extractLena",extractLena)#显示解密解码后的图像
cv2.waitKey() #等待按键
cv2.destroyAllWindows()#按键后关闭刚显示的所有窗口
```

五、实验结果




key
lenaXorKey
encryptFace


noFace1
maskFace
extractOriginal


extractFace
noFace2
extractLena

图像空域滤波

一、实验目的

（1）理解空域滤波的概念、作用；

（2）了解空域滤波的类型：线性滤波和非线性滤波；

（3）掌握常用的线性滤波原理、特点及实现方法，包括均值滤波、方框滤波、高斯滤波等；

（4）掌握常用的非线性滤波原理、特点及实现方法，如中值滤波，双边滤波等。

二、实验内容

空域滤波是一种邻域处理方法，通过直接在图像空间中对邻域内像素进行处理，达到平滑或锐化图像的作用。此外，在图像识别中，通过滤波还可以抽出图像的特征作为图像识别的特征模式。

空域滤波是图像处理领域中广泛使用的主要工具。空域滤波主要可以分为线性滤波和非线性滤波，线性滤波和频域滤波存在一一对应的关系。空域滤波可以用于非线性滤波，但是频域滤波却不能用于非线性滤波。

编写 Python 程序，采用中值滤波、均值滤波、高斯滤波三种滤波方式分别对含噪图像进行降噪处理，显示原图像及滤波后的图像，并对实验结果进行分析。

三、实验原理

平滑处理(smoothing)也称模糊处理(bluring)，是一种简单且使用频率很高的图像处理方法。平滑处理的用途有很多，最常见的是用来减少图像上的噪点或者失真。在降低图像分辨率时，平滑处理是非常好用的方法。

图像滤波指在尽量保留图像细节特征的条件下对目标图像的噪声进行抑制，是图像预处理中不可缺少的操作，其处理效果的好坏将直接影响后续图像处理和分析的有效性和可靠性。

消除图像中的噪声成分叫作图像的平滑化或滤波操作。信号或图像的能量大部分集中在幅度谱的低频和中频段，而在较高频段，有用的信息经常被噪声淹没。因此一个能降低高频成分幅度的滤波器就能够减弱噪声的影响。

图像滤波的目的有两个:即抽出对象的特征作为图像识别的特征模式；或为适应图像处理的要求，消除图像数字化时所混入的噪声。而对滤波处理的要求也有两条，即不能损坏图像的轮廓及边缘等重要信息，同时要使图像清晰视觉效果好。

平滑滤波是低频增强的空间域滤波技术。根据其目的分为两类:一类是模糊，另一类是消除噪音。

空间域的平滑滤波一般采用简单平均法进行，就是求邻近像元点的平均亮度值。邻

域的大小与平滑的效果直接相关，邻域越大平滑的效果越好，但邻域过大，平滑也会使边缘信息损失的越大，从而使输出的图像变得模糊，因此需合理选择邻域的大小。

滤波器类似一个包含加权系数的窗口，当使用这个滤波器平滑处理图像时，就把这个窗口放到图像之上，透过这个窗口来看我们得到的图像。

滤波器的种类有很多，在 OpenCV 中，提供了如下 5 种常用的图像平滑处理操作方法，它们分别被封装在单独的函数中，使用起来非常方便。

（1）方框滤波——BoxBlur 函数；

（2）均值滤波(邻域平均滤波)——Blur 函数；

（3）高斯滤波——GaussianBlur 函数；

（4）中值滤波——medianBlur 函数；

（5）双边滤波——bilateralFiter 函数。

（一）线性滤波

线性滤波器经常用于剔除输入信号中不想要的频率成分或者从许多频率中选择一个想要的频率成分。

几种常见的线性滤波器如下:

（1）低通滤波器:允许低频率成分通过；

（2）高通滤波器:允许高频率成分通过；

（3）带通滤波器:允许一定范围频率成分通过；

（4）带阻滤波器:阻止一定范围频率成分通过并且允许其他频率成分通过；

（5）全通滤波器:允许所有频率成分通过，仅仅改变相位关系；

（6）陷波滤波器(Band-Stop Filter):阻止一个狭窄频率范围成分通过，是一种特殊带阻滤波器。

滤波是将信号中特定波段频率成分滤除的操作，是抑制和防止干扰信息的一项重要措施。以高斯滤波为例，滤波可分低通滤波和高通滤波两种:高斯滤波是指用高斯函数作为滤波函数的滤波操作，是否模糊，取决于是高斯低通还是高斯高通，低通就是模糊，高通就是锐化。即高斯滤波是指用高斯函数作为滤波函数的滤波操作，而高斯模糊就是高斯低通滤波。

下图中（a）图像与(b)图像的卷积产生(c)图像。目标图像(c)中红色标记的像素是利用原图像（a）中红色标记的像素计算得到的。

60	98	127	132	133	137	133
65	98	123	126	128	131	133
65	96	115	119	123	135	137
63	91	107	113	122	138	134
59	80	97	110	123	133	134
53	68	83	97	113	128	133
116	102	84	70	58	50	50
50	52	58	69	86	101	120

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

（a）$f(x,y)$　　（b)$h(x,y)$　　（c）$g(x,y)$

线性滤波处理的输出像素值 g(i,j)是输入像素值 $f(i+k, j+l)$的加权和，如下：

$$g(i,j)=\sum_{k,l} f(i+k,j+l)h(k,l)$$

其中 $h(k,l)$被称为“核”，是滤波器的加权系数，即滤波器的“滤波系数”。

上面的式子可以简单写作：

$$g = f*h$$

其中：f 表示输入像素值，h 表示加权系数“核”，g 表示输出像素值。

在 OpenCV 中，提供了如下三种常用的线性滤波操作，它们分别被封装在单独的函数中，使用起来非常方便。

（1）方框滤波——boxblur 函数；

（2）均值滤波——blur 函数；

（3）高斯滤波——GassianBlur 函数；

1.方框滤波

方框滤波(box Filtr)被封装在 boxblur 的函数中，即 boxblur 函数的作用是使用方框滤波器(box filter)来模糊一张图片，从 src 输入，从 dst 输出。

在 Opencv 的源函数定义过程中，函数原型如下：

cv2.boxFilter (src, ddepth, ksize，anchor,，normalize, borderType)

src 是需要处理的图像，即原始图像，它可以有任意数量的通道，且对通道是独立处理的，可以处理任意通道数的图片。但需要注意，待处理的图片深度为 CV_ 8U、CV_ 16U、CV_ 16S、CV_32F 以及 CV_64F 等类型之一。

ddepth 是处理结果图像的深度，其中取值为-1 代表使用原图深度，

ksize 是滤波核大小是指在滤波处理过程中所选择的邻域图像的高度和宽度，例如，滤波核的值可以为(3,3),表示以 3×3 大小的邻域均值作为图像均值滤波处理的结果

anchor 表示锚点(即被平滑的那个点)。如果这个点坐标是负值的话，表示取核的中心为锚点，即系统默认值 Point(-1,-1)表示这个锚点在核的中心。在特殊情况下可以指定不同的点作为锚点。

normalize 表示在滤波时是否进行归一化（这里指将计算结果规范化为当前像素值范围内的值）处理，该参数是一个逻辑值，可能为真，可能为假。

当参数 normalize=1 时，表示要进行归一化处理，即用邻域像素值的和除以面积。

当参数 normalize=0 时，表示不需要进行归一化处理，直接使用邻域像素值的和。

borderType 表示边界样式，决定以何种方式处理边界。

通常情况下，在使用方框滤波函数时，对于参数 anchor，normalize 和 borderType，直接采用其默认值即可。因此，函数 cv2.boxFilter()的常用形式为：

dst = cv2.boxFilter(src, ddepth, ksize)

BoxFilter()函数方框滤波所用的核表示如下：

$$K = a\begin{bmatrix} 111 & \dots & 111 \\ 111 & \dots & 111 \\ 111 & \dots & 111 \end{bmatrix}$$

其中：

$$\alpha = \begin{cases} \dfrac{1}{ksize.width * ksize.height}, & normalize = 1 \\ 1, & normalize = 0 \end{cases}$$

当 normalize= true 的时候，方框滤波就变成了我们熟悉的均值滤波。也就是说，均值滤波是方框滤波归一化(normalized)后的特殊情况。其中，归一化就是把要处理的量都缩放到一个范围内，比如(0,1)， 以便统一处理和直观量化。而非归一化(Unnormalized) 的方框滤波用于计算每个像素邻域内的积分特性，比如密集光流算法(dense optical flow algorithms)中用到的图像倒数的协方差矩阵(covariance matrices of image derivatives)。

如果要在可变的窗口中计算像素总和，可以使用 integral0 函数。

均值滤波是最简单的一种滤波操作，输出图像的每一个像素是核窗口内输入图像对应像素的平均值(所有像素加权系数相等)，其实说白了它就是归一化后的方框滤波。blur 函数内部中其实就是调用了一下 boxFilter。

2.均值滤波

均值滤波是典型的线性滤波算法，主要方法为邻域平均法，即用一片图像区域的各个像素的均值来代替原图像中的各个像素值。一般需要在图像上对目标像素给出一个模板(内核)，该模板包括了其周围的邻近像素(例如以目标像素为中心的周围 8 个像素，构成一个滤波模板，即假设 3*3 窗口，去掉目标像素本身)。再用模板中的全体像素的平均值来代替原来像素值。即对待处理的当前像素点(x,y)选择一个模板，该模板由其近邻的若干像素组成，求模板中所有像素的均值，再把该均值赋予当前像素点(x,y)，作为处理后图像在该点上的灰度点$g(x,y)$,即 $g(x,y) = \frac{1}{m}\sum f(x,y)$ 其中 m 为该模板中包含当前像素在内的像素总个数。

均值滤波本身存在着固有的缺陷，即它不能很好地保护图像细节，在图像去噪的同时也破坏了图像的细节部分，从而使图像变得模糊，不能很好地去除噪声点。

在 OpenCV 中使用均值滤波——blur 函数，blur 函数的作用是:对输入的图像 src 进行均值滤波后通过 dst 输出。blur 函数在 OpenCV 官方文档中，给出的核如下:

$$K = \frac{1}{ksize.width * ksize.height}\begin{bmatrix} 111...11 \\ 111...11 \\ ... \\ 111...11 \end{bmatrix}$$

这个内核显然是在求均值，即 blur 函数封装的就是均值滤波。blur 函数的原型如下：

Opencv：dst=cv2.blurr(src, ksize, anchor, borderType_)

src 是需要处理的图像，即原始图像，它可以有任意数量的通道，且对通道是独立处理的，可以处理任意通道数的图像。但需要注意，待处理的图片深度为 CV_ 8U、CV_ 16U、CV_ 16S、CV_32F 以及 CV_64F 等类型之一。

ksize 是滤波核大小是指在滤波处理过程中所选择的邻域图像的高度和宽度，例如滤波核的值可以为(5,5)，表示以 5×5 大小的邻域均值作为图像均值滤波处理的结果。

anchor 表示锚点(即被平滑的那个点)。如果这个点坐标是负值，表示取核的中心为锚点，即系统默认值 Point(-1,-1)表示这个锚点在核的中心。在特殊情况下可以指定不同的点作为锚点。

borderType 是边界样式，决定以何种方式处理边界。

通常情况下，在使用方框滤波函数时，对于参数 anchor 和 borderType，直接采用其默认值即可。因此，函数 cv2.boxFilter()的常用形式为：

dst = cv2.blur(src, ksize)

3.高斯滤波

（1）高斯滤波的理论简析

高斯滤波是一种线性平滑滤波，可以消除高斯噪声，广泛应用于图像处理的减噪过程。通俗地讲，高斯滤波就是对整幅图像进行加权平均的过程，每一个像素点的值，都由其本身和邻域内的其他像素值经过加权平均后得到。高斯滤波的具体操作是用一个模板(或称卷积、掩模)扫描图像中的每一个像素，用模板确定的邻域内像素的加权平均灰度值去替代模板中心像素点的值。

大家常说高斯滤波是最有用的滤波操作，虽然它用起来效率往往不是最高的。高斯模糊技术生成的图像，其视觉效果就像是经过一个半透明屏幕在观察图像，这与镜头焦外成像效果散景以及普通照明阴影中的效果明显不同。高斯平滑也用于计算机视觉算法中的预处理阶段，以增强图像在不同比例大小下的图像效果。从数学的角度来看，图像的高斯模糊过程就是图像与正态分布做卷积。由于正态分布又叫作高斯分布，所以这项技术就叫作高斯模糊。

图像与圆形方框模糊做卷积将会生成更加精确的焦外成像效果。由于高斯函数的傅里叶变换是另外一个高斯函数，所以高斯模糊对于图像来说就是一个低通滤波操作。高斯滤波器是一类根据高斯函数的形状来选择权值的线性平滑滤波器。高斯平滑滤波器对于抑制服从正态分布的噪声非常有效。一维零均值高斯函数如下：

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

其中，高斯分布参数σ 决定了高斯函数的宽度。对于图像处理来说，常用二维零均值高斯函数作平滑滤波器。

二维高斯函数如下：

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

（2）高斯滤波 GaussianBlur 函数

GaussianBlur 函数的作用是用高斯滤波器来模糊一张图片，对输入的图 src 进行高斯滤波通过 dst 输出，它是将源图像和指定的高斯核函数作卷积运算。

Opencv：dst = cv2.GaussianBlur(src, ksize, sigmaX, sigmaY,borderType)

dst 是返回值，表示进行高斯滤波后得到的处理结果。

src 是需要处理的图像，即原始图像。它可以有任意数量的通道，并能对各个通道独立处理。图像深度应该是 CV_8U、CV_16U、CV_16S、CV_32F 或者 CV_64F 中的一种 ksize 是滤波核的大小。滤波核大小是指在滤波处理过程中其邻域图像的高度和宽度需要注意，滤波核的值必须是奇数。

sigmaX 是卷积核在水平方向上(X 轴方向)的标准差，其控制的是权重比例，不同的 sigmaX 决定的卷积核，它们在水平方向上的标准差不同

sigmaY 是卷积核在垂直方向上(Y 轴方向)的标准差。如果将该值设置为 0，则只采用 sigmaX 的值；如果 sigmaX 和 sigmaY 都是 0，则通过 ksizewidth 和 ksize.height 计算得到。

其中:

sigmaX=0.3×[(ksize.widh-1)×0.5-1]+0.8

sigmaY=03×[(ksize.height-1)×0.5-1]+0.8

brder 是边界样式，决定以何种方式处理边界，一般情况下,不需要考虑值，直接采用默认值即可。

在该函数中，sigmaY 和 borderType 是可选参数和 sigmaX 是必选参数，但是可以将该参数设置为 0，让函数自己去计算 sigmaX 的具体值。

建议显式地指定 ksize.sigmaX 和 sigmaY 三个参数的值，以避免将来函数修改后可能造成的语法错误。当然，在实际处理中，可以显式指定 sigmaX 和 sigmaY 为默认值 0. 因此，函数 cv2.GaussianBlur 的常用形式为:

dst = cv2.GaussianBlur(src,ksize,0, 0)

（二）非线性滤波

线性滤波中两个信号之和的响应与它们各自响应之和相等。换句话说，每个像素的输出值是一些输入像素的加权和。线性滤波器易于构造，并且易于从频率响应角度来进行分析。然而，在很多情况下，使用邻域像素的非线性滤波会得到更好的效果。比如在噪声是散粒噪声而不是高斯噪声，即图像偶尔会出现很大的值的时候，用高斯滤波器对图像进行模糊的话，噪声像素是不会被去除的，它们只是转换为更为柔和但仍然可见的散粒，此时用非线性滤波会可得到更好的效果，下面介绍两种非线性滤波器。

1.中值滤波

中值滤波(Median fiter)是一种典型的非线性滤波技术，基本思想是用像素点邻域灰度值的中值来代替该像素点的灰度值，该方法在去除脉冲噪声、椒盐噪声的同时又能保留图像的边缘细节。

中值滤波是基于排序统计理论的一种能有效抑制噪声的非线性信号处理技术，其基本原理是把数字图像或数字序列中一点的值用该点的一一个邻域中各点值的中值代替，让周围的像素值接近真实值，从而消除孤立的噪声点。这对于斑点噪声(speckle noise) 和椒盐噪声(salt-and-pepper noise)来说尤其有用，因为它不依赖于邻域内那些与典型值差别很大的值。中值滤波器在处理连续图像窗函数时与线性滤波器的工作方式类似，但滤波过程却不再是加权运算。

中值滤波在一定的条件下可以克服常见线性滤波器，如最小均方滤波、方框滤波器、均值滤波等带来的图像细节模糊，而且对滤除脉冲干扰及图像扫描噪声非常有效，也常用于保护边缘信息。保存边缘的特性使它在不希望出现边缘模糊的场合也很有用，是非常经典的平滑噪声处理方法。

中值滤波与均值滤波器比较:

优势:在均值滤波器中，由于噪声成分被放入平均计算中，所以输出受到了噪声的影响。但是在中值滤波器中，由于噪声成分很难选上，所以几乎不会影响到输出。因此同样用 3*3 区域进行处理，中值滤波消除噪声能力更胜一筹。中值滤波无论是在消除噪声还是保存边缘方面都是一个不错的方法。

劣势:中值滤波花费的时间是均值滤波的 5 倍以上。

顾名思义，中值滤波选择每个像素的邻域像素中的中值作为输出，或者说中值滤波将每一像素点的灰度值设置为该点某邻域窗口内的所有像素点灰度值的中值。例如，取 3*3 的函数窗，计算以点$[i, j]$为中心的函数窗像素中值，具体步骤如下：

（1）按强度值大小排列像素点。

（2）选择排序像素集的中间值作为点$[i, j]$的新值。

一般采用奇数点的邻域来计算中值，但像素点数为偶数时，中值就取排序像素中间两点的平均值。中值滤波在一定条件下，可以克服线性滤波器(如均值滤波等)所带来的图像细节模糊，对滤除脉冲干扰即图像扫描噪声最为有效，而且在实际运算过程中并不需要图像的统计特性，也给计算带来不少方便。但是对一些细节(特别是细、尖顶等)多的图像不太适合。

2.双边滤波

双边滤波(Bilateral filter) 是-种非线性的滤波方法，是结合图像的空间邻近度和像素值相似度的一种折中处理，同时考虑空域信息和灰度相似性，达到保边去噪的目的，具有简单、非迭代、局部的特点。

双边滤波器的好处是可以做边缘保存(edge preserving)。 以往常用维纳滤波或者高斯滤波降噪，但二者都会较明显地模糊边缘，对于高频细节的保护效果并不明显。双边滤波器顾名思义，比高斯滤波多了一个高斯方差σ_d，它是基于空间分布的高斯滤波函数，所以在边缘附近，离得较远的像素不会对边缘上的像素值影响太多，这样就保证了边缘附近像素值的保存。但是，由于保存了过多的高频信息，对于彩色图像里的高频噪声，双边滤波器不能够干净地滤掉，只能对于低频信息进行较好地滤波。

在双边滤波器中，输出像素的值依赖于邻域像素值的加权值组合，公式如下：

$$g(x,y)=\frac{\sum_{k,l} f(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

而加权系数$w(i,j,k,l)$取决于定义域核和值域核的乘积。其中定义域核表示如下：

$$d(i,j,k,l)=\exp(-\frac{(i-k)^2+(j-l)^2}{2\sigma_r^2})$$

值域核表示如下：

$$r(i,j,k,l)=e^{-\frac{\|f(i,j)-f(k,l)\|^2}{2\sigma_r^2}}$$

两者相乘后，就会产生依赖于数据的双边滤波权重函数：

$$w(i,j,k,l)=e^{-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2}-\frac{\|f(i,j)-f(k,l)\|^2}{2\sigma_r^2}}$$

四、实验步骤:

1.运行 jupyter lab

（1）在“实验”文件夹空白处单击鼠标右键，选择“打开终端”，在终端界面输入“jupyter lab”。

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器。

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为所做实验名。

2.利用中值滤波器处理图像

（1）导入库函数 cv2、numpy。

参考代码:

```
import numpy as np
import cv2
```

（2）定义中值滤波函数 median_filter(img, K_size=3)，模板大小为 3*3。

参考代码:

```
# 定义中值滤波函数
def median_filter(img, k_size=3):
    h, w, c = img.shape
    # 边缘补零
    pad = k_size // 2
    out = np.zeros((h + pad * 2, w + pad * 2, c), dtype=np.float)
    out[pad:pad + h, pad:pad + w] = img.copy().astype(np.float)
    tmp = out.copy()
    # 滤波
    for y in range(h):
        for x in range(w):
            for c in range(c):
                out[pad + y, pad + x, c] = np.median(tmp[y:y + k_size, x:x + k_size, c])
    out = out[pad:pad + h, pad:pad + w].astype(np.uint8)
    return out
```

（3）利用 cv2.imread()函数读取原始图片 3gong.jpg。调用中值滤波函数滤波，处理结果保存至 out1。

参考代码:

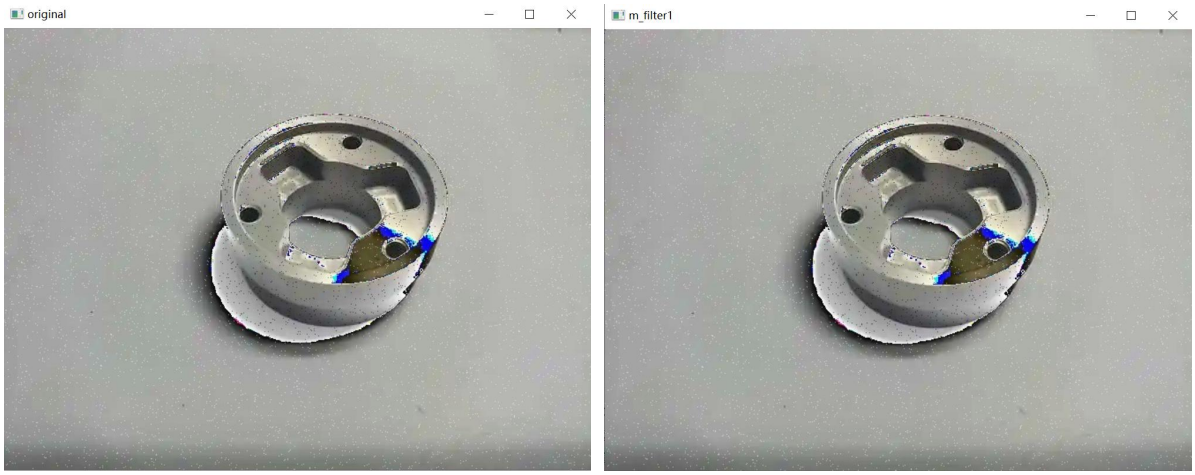
```
# 读取原始图像，调用中值滤波函数滤波
img1 = cv2.imread("/home/retoo/Desktop/实验/数据集/4.数字图像处理/3gong.jpg")
out1 = median_filter(img1, k_size=3)
```

（4）显示原图像和中值滤波后图像。

参考代码:

```
# 显示中值滤波结果
cv2.imshow("original", img1)
cv2.imshow("m_filter1", out1)
cv2.waitKey()  # 等待按键
cv2.destroyAllWindows()  # 按键后关闭刚显示的所有窗口
```

中值滤波运行结果如下图所示。



（a）原图　　（b）中值滤波后

中值滤波效果图

3.利用均值滤波器处理图像

（1）导入库函数 cv2、numpy。

参考代码:

```
import numpy as np
import cv2
```

（2）定义均值滤波函数 average_filter(img, G=3)，模板大小为 3*3。

参考代码:

```
# 定义均值滤波函数
def average_filter(img, g=3):
    out = img.copy()
    h, w, c = img.shape
    nh = int(h / g)
    nw = int(w / g)
    for y in range(nh):
        for x in range(nw):
            for m in range(c):
                out[g * y:g * (y + 1), g * x:g * (x + 1), m] = np.mean(
```

```
                out[g * y:g * (y + 1), g * x:g * (x + 1), m]).astype(np.int)
    return out
```

（3）利用 cv2.imread()函数读取原始图片 gong.jpg，再调用均值滤波函数滤波，处理结果保存至 out2。

参考代码:

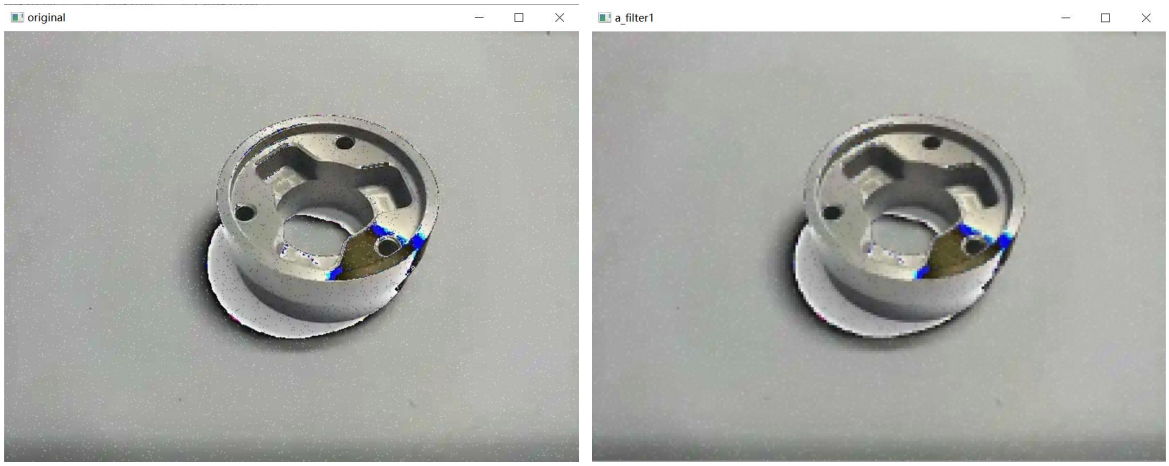
```
# 读取原始图像，调用均值滤波函数滤波
img2 = cv2.imread("/home/retoo/Desktop/实验/数据集/4.数字图像处理/3gong.jpg")
out2 = average_filter(img2, g=3)
```

（4）显示原图像和均值滤波后图像。

参考代码:

```
# 显示均值滤波结果
cv2.imshow("original", img2)
cv2.imshow("a_filter1", out2)
cv2.waitKey()  # 等待按键
cv2.destroyAllWindows()  # 按键后关闭刚显示的所有窗口
```

均值滤波运行结果如下图所示。



（a）原图　　（b）均值滤波后

图 3 均值滤波效果图

4. 利用高斯滤波器处理图像

（1）导入库函数 cv2、numpy。

参考代码:

```
import numpy as np
import cv2
```

（2）定义高斯滤波函数 gaussian_filter(img, k_size=3, sigma=1.3)，模板大小为 3*3。

首先获取图像大小，接着对图像进行边缘补零(zero padding)处理，再根据高斯滤波器的核大小和标准差大小实现高斯滤波。

参考代码:

```
# 定义高斯滤波函数
def gaussian_filter(img, k_size=3, sigma=1.3):
    if len(img.shape) == 3:
        h, w, c = img.shape
    else:
        img = np.expand_dims(img, axis=-1)
        h, w, c = img.shape
    # 边缘补零
    pad = k_size // 2
    out = np.zeros((h + pad * 2, w + pad * 2, c), dtype=np.float)
    out[pad: pad + h, pad: pad + w] = img.copy().astype(np.float)
    # 准备核
    k = np.zeros((k_size, k_size), dtype=np.float)
    for x in range(-pad, -pad + k_size):
        for y in range(-pad, -pad + k_size):
            k[y + pad, x + pad] = np.exp(-(x ** 2 + y ** 2) / (2 * (sigma ** 2)))
    k /= (2 * np.pi * sigma * sigma)
    k /= k.sum()
    tmp = out.copy()
    # 滤波
    for y in range(h):
        for x in range(w):
            for m in range(c):
                out[pad + y, pad + x, m] = np.sum(k * tmp[y: y + k_size, x: x + k_size, m])
    out = np.clip(out, 0, 255)
    out = out[pad: pad + h, pad: pad + w].astype(np.uint8)
    return out
```

（3）利用 cv2.imread()函数读取原始图片 gong.jpg，再调用高斯滤波函数滤波，处理结果保存至 out3。

参考代码:

```
# 读取原始图像，调用高斯滤波函数滤波
img3 = cv2.imread("/home/retoo/Desktop/实验/数据集/4.数字图像处理/3gong.jpg")
out3 = gaussian_filter(img3, k_size=3, sigma=1.3)
```

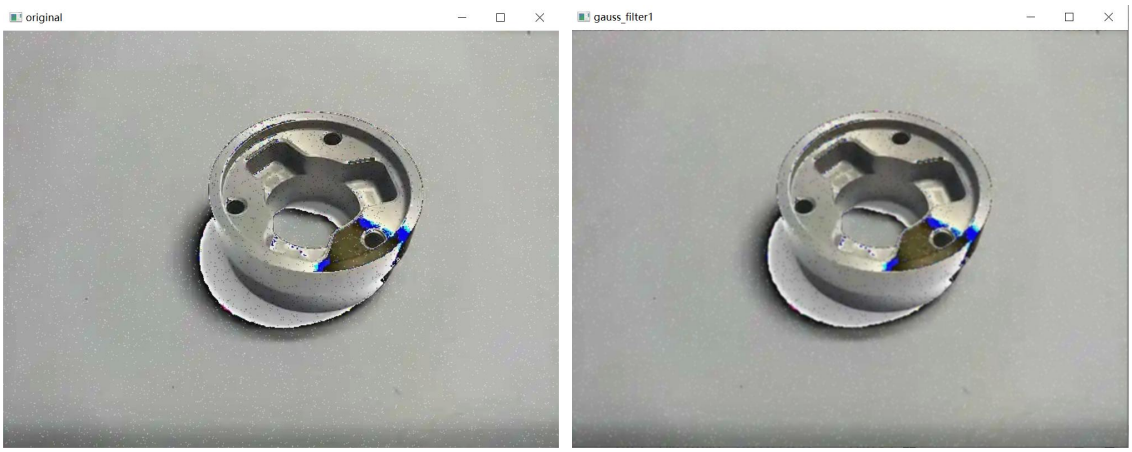
（4）显示原图像和高斯滤波后图像。

参考代码:

```
# 显示均值滤波结果
cv2.imshow("original", img3)
cv2.imshow("gauss_filter1", out3)
cv2.waitKey()  # 等待按键
```

```
cv2.destroyAllWindows()  # 按键后关闭刚显示的所有窗口
```

高斯滤波运行结果如下图所示。



（a）原图　　（b）高斯滤波后

图 4 高斯滤波效果图

五、实验结果分析

对同一个含噪声的图片分别采用中值滤波、均值滤波、高斯滤波后，其中均值滤波和高斯滤波明显降低了图像的噪声，图片变模糊了，但高斯滤波运行时间长。

视觉系统认知

一、实验目的

（1）了解 2D 和 3D 视觉系统硬件布局。

（2）熟悉 2D 和 3D 视觉系统的硬件参数；

二、实验内容

本实验要求在 2D 和 3D 视觉系统硬件结构的基础上，掌握系统的硬件系统原理和端口资源分配，通过运行演示程序，了解系统的功能。

三、实验环境

硬件环境	JetsonNX
操作系统	Linux
实验设备	2D 相机，3D 相机，Jetson NX
实验配件	相机线

四、实验原理

1. 2D 和 3D 视觉系统认知

2D 和 3D 视觉系统硬件布局如图所示，其包含部件如下：

（1）2D 视觉相机；

（2）3D 相机与云台系统；

2D 视觉和 3D 视觉与云台系统

2. 视觉传感系统硬件参数

RealsenseD415 的硬件包含了两个深度相机，一个 RGB 相机和一个结构光红外投影仪。深度卷帘相机（逐行扫描），红外结构光深度测距。

表 1：3D 视觉系统参数

Realsense D415 参数表

深度范围（米）	0.3m - 10m
功耗	360mW
彩色传感器	OV2740
彩色图分辨率	1920 x 1080 at 30 fps
深度图分辨率	Up to 1280 x 720 Up to 90 fps Rolling Shutter
彩色 FOV	69.4° x 42.5° x 77° (+/- 3°) (HVD)
深度 FOV	65°±2° x 40°±1° x 72°±2° (HVD)
快门	深度：卷帘 RGB：卷帘
数据传输	USB-C* 3.1 Gen 1
支持 ROS	支持
平台	Linux/Windows/Mac
供电方式	USB-C
原理	Active IR stereo
使用环境	室内、室外
基线 mm	55
尺寸（毫米）	99 mm x 20 mm x 23 mm
重量 g	75

表 2：云台系统参数

云台	
旋转角度范围	0-180°
控制方式	串口/波特率：115200，无校验
接口协议	0xFF, 0xFE, 0x02, 0x01, 0x00, 0x5A, 0x5A, 0x00, 0x0D, 0x0A （从 0 位开始，第五位，第六位分别表示舵机 1 和舵机 2 转动的 90 度）

表 3：2D 视觉系统参数

2D 视觉系统

分辨率	640*480
图像格式	YUV
帧率	30fps
镜头	F3. 2mm/58°
尺寸	19mm*24mm*18mm
白平衡	自动
曝光	自动
工作电流	<200mA
工作温度	0 C--+50 C

五、实验步骤

将视觉系统的 USB 接入 JetsonNano 处理器上(3D 视觉默认已接好，2D 视觉需要手动插入 USB)，云台系统默认已接到控制板，只需要通过串口发送指令控制即可，在后面的机器视觉实验中使用 USB 相机或者 3D RGB 相机进行图像采集，采集程序如下所示，USB 相机采集程序对机器视觉实验的图像采集都适用，包括相机标定部分的图像采集程序是一致的。

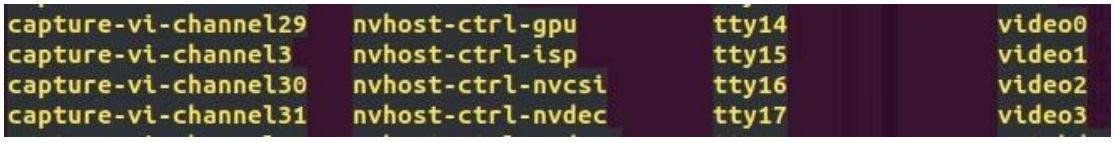
点击打开摄像头，程序会自动打开相机，点击拍照，会采集当前的照片并保存到 pic 文件夹下。保存的文件夹名称及路径可以在程序中进行自定义修改。

在使用 3D 相机或者 USB 相机进行彩色图像采集时只需要将程序中打开相机的 ID 改为对应的相机端口即可。例如：

self.cap = cv2.VideoCapture() # 准备获取图像

self.CAM_NUM =0#定义 USB 端口号，由于系统对 Video 设备初始化端口是按照随机先后顺序来的，如果没有插 2D 相机，插了 3D 相机，那么 3D 相机三个摄像头对应的是 0,1,2，这个时候插上 2D 相机那么 2D 相机的端口号是 3。同理如果插了 2D，没插 3D 那么 2D 的端口号是 0。需要根据指令“ls /dev”来查看对应的设备的端口号是多少。

flag = self.cap.open(self.CAM_NUM)#打开对应端口号的相机



图像采集程序

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终

端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）在单元格输入“%load 相机采集程序.py”,点击运行按钮即可将程序载入到单元格。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

相机采集程序 UI 基于 pyQt5 进行编写。

参考代码：

```
from PyQt5 import QtCore, QtGui, QtWidgets
from sys import argv, exit
from PyQt5. QtWidgets import QApplication, QMainWindow
import time
import cv2
```

3.界面 UI：UI 界面的一些布局设置

参考代码：

```
class Ui_MainWindow(object):
    def __init__(self, MainWindow):
        self.timer_camera = QtCore.QTimer() # 定时器
        self.setupUi(MainWindow)
        self.retranslateUi(MainWindow)
        self.cap = cv2.VideoCapture(0)  # 准备获取图像
        ret, image = self.cap.read()
        if ret != True:
            self.CAM_NUM = 3 #由于系统初始化 USB 口是随机顺序,USB 相机的端口号可能是 0 或者 3
        else:
            self.CAM_NUM = 0
        self.slot_init() # 设置槽函数
    #界面设置
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.setWindowModality(QtCore.Qt.NonModal)
        MainWindow.resize(765, 645)
        MainWindow.setMinimumSize(QtCore.QSize(765, 645))
        MainWindow.setMaximumSize(QtCore.QSize(16777215, 16777215))
        MainWindow.setToolTip("")
```

```
        MainWindow.setAutoFillBackground(False)
        MainWindow.setTabShape(QtWidgets.QTabWidget.Rounded)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout(self.centralwidget)
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")
        self.label = QtWidgets.QLabel(self.centralwidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())
        self.label.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("华文隶书")
        font.setPointSize(20)
        self.label.setFont(font)
        self.label.setAlignment(QtCore.Qt.AlignCenter)
        self.label.setObjectName("label")
        self.verticalLayout.addWidget(self.label)
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setSizeConstraint(QtWidgets.QLayout.SetDefaultConstraint)
        self.horizontalLayout.setContentsMargins(-1, 50, -1, -1)
        self.horizontalLayout.setSpacing(0)
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.pushButton_open = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_open.setMinimumSize(QtCore.QSize(100, 40))
        self.pushButton_open.setMaximumSize(QtCore.QSize(120, 40))
        font = QtGui.QFont()
        font.setFamily("华文彩云")
        font.setPointSize(12)
        self.pushButton_open.setFont(font)
        self.pushButton_open.setObjectName("pushButton_open")
        self.horizontalLayout.addWidget(self.pushButton_open)
        self.pushButton_take = QtWidgets.QPushButton(self.centralwidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.pushButton_take.sizePolicy().hasHeightForWidth())
        self.pushButton_take.setSizePolicy(sizePolicy)
        self.pushButton_take.setMinimumSize(QtCore.QSize(100, 40))
```

```
        self.pushButton_take.setMaximumSize(QtCore.QSize(100, 40))
        font = QtGui.QFont()
        font.setFamily("华文彩云")
        font.setPointSize(12)
        self.pushButton_take.setFont(font)
        self.pushButton_take.setObjectName("pushButton_take")
        self.horizontalLayout.addWidget(self.pushButton_take)
        self.pushButton_close = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_close.setMinimumSize(QtCore.QSize(100, 40))
        self.pushButton_close.setMaximumSize(QtCore.QSize(130, 40))
        font = QtGui.QFont()
        font.setFamily("华文彩云")
        font.setPointSize(12)
        self.pushButton_close.setFont(font)
        self.pushButton_close.setObjectName("pushButton_close")
        self.horizontalLayout.addWidget(self.pushButton_close)
        self.horizontalLayout.setStretch(0, 1)
        self.horizontalLayout.setStretch(1, 1)
        self.horizontalLayout.setStretch(2, 1)
        self.verticalLayout.addLayout(self.horizontalLayout)
        self.label_face = QtWidgets.QLabel(self.centralwidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Expanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_face.sizePolicy().hasHeightForWidth())
        self.label_face.setSizePolicy(sizePolicy)
        self.label_face.setMinimumSize(QtCore.QSize(0, 0))
        self.label_face.setMaximumSize(QtCore.QSize(16777215, 16777215))
        font = QtGui.QFont()
        font.setFamily("楷体")
        font.setPointSize(16)
        self.label_face.setFont(font)
        self.label_face.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.label_face.setStyleSheet("background-color: rgb(192, 218, 255);")
        self.label_face.setAlignment(QtCore.Qt.AlignCenter)
        self.label_face.setObjectName("label_face")
        self.verticalLayout.addWidget(self.label_face)
        self.verticalLayout.setStretch(2, 5)
        self.horizontalLayout_2.addLayout(self.verticalLayout)
        MainWindow.setCentralWidget(self.centralwidget)
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
    #界面 UI 控件
```

```
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Qt-Camera"))
        self.label.setText(_translate("MainWindow", "图像采集程序"))
        self.pushButton_open.setToolTip(_translate("MainWindow", "点击打开摄像头"))
        self.pushButton_open.setText(_translate("MainWindow", "打开摄像头"))
        self.pushButton_take.setToolTip(_translate("MainWindow", "点击拍照"))
        self.pushButton_take.setText(_translate("MainWindow", "拍照"))
        self.pushButton_close.setToolTip(_translate("MainWindow", "点击关闭摄像头"))
        self.pushButton_close.setText(_translate("MainWindow", "关闭摄像头"))
        self.label_face.setText(_translate("MainWindow", "<html><head/><body><p
align=\"center\"><img src=\":/newPrefix/pic/Hint.png\"/><span style=\" font-size:28pt;\">点击打
开摄像头</span><br/></p></body></html>"))
```

4.槽函数设置 ：将 UI 控件与事件函数进行绑定

参考代码：

```
# 设置槽函数
def slot_init(self):
    self.pushButton_open.clicked.connect(self.button_open_camera_click)
    self.timer_camera.timeout.connect(self.show_camera)
    self.pushButton_close.clicked.connect(self.closeEvent)
    self.pushButton_take.clicked.connect(self.takePhoto)
```

5.事件函数:打开相机，关闭相机，采集图像等函数

参考代码：

```
#定时器事件函数
    def button_open_camera_click(self):
        if self.timer_camera.isActive() == False:
            flag = self.cap.open(self.CAM_NUM)
            if flag == False:
                msg = QtWidgets.QMessageBox.warning(
                    self, u"Warning", u"请检测相机与电脑是否连接正确",
                    buttons=QtWidgets.QMessageBox.Ok,
                    defaultButton=QtWidgets.QMessageBox.Ok)
            else:
                self.timer_camera.start(30)
    #显示图像
    def show_camera(self):
        flag, self.image = self.cap.read()
        show = cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB)#图像格式转为 RGB
        showImage = QtGui.QImage(show.data, show.shape[1], show.shape[0],
QtGui.QImage.Format_RGB888)#设置显示格式
        self.label_face.setPixmap(QtGui.QPixmap.fromImage(showImage))#显示图像
```

```
            self.label_face.setScaledContents(True)#图像自适应窗口大小
        #采集图像并保存
        def takePhoto(self):
            if self.timer_camera.isActive() != False:
                now_time =
time.strftime('%Y-%m-%d-%H-%M-%S', time.localtime(time.time()))#获取系统时间
                print(now_time)
                cv2.imwrite('pic/pic_'+str(now_time)+'.jpg', self.image)#保存图片
                cv2.putText(self.image, 'The picture have saved !',
                            (int(self.image.shape[1]/2-130),
int(self.image.shape[0]/2)),
                            cv2.FONT_HERSHEY_SCRIPT_COMPLEX,
                            2.0, (255, 0, 0), 1)
                show = cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB)  #BGR 转 RGB
                showImage = QtGui.QImage(show.data, show.shape[1], show.shape[0],
QtGui.QImage.Format_RGB888)
                self.label_face.setPixmap(QtGui.QPixmap.fromImage(showImage))
                self.label_face.setScaledContents(True)
        #关闭相机
        def closeEvent(self):
            if self.timer_camera.isActive() != False:
                ok = QtWidgets.QPushButton()
                cacel = QtWidgets.QPushButton()

                msg = QtWidgets.QMessageBox(QtWidgets.QMessageBox.Warning, u"关闭",
u"是否关闭！")

                msg.addButton(ok,QtWidgets.QMessageBox.ActionRole)
                msg.addButton(cacel, QtWidgets.QMessageBox.RejectRole)
                ok.setText(u'确定')
                cacel.setText(u'取消')

                if msg.exec_() != QtWidgets.QMessageBox.RejectRole:

                    if self.cap.isOpened():
                        self.cap.release()
                    if self.timer_camera.isActive():
                        self.timer_camera.stop()
                    self.label_face.setText("<html><head/><body><p align=\"center\"><img
src=\":/newPrefix/pic/Hint.png\"/><span style=\" font-size:28pt;\">点击打开摄像头
</span><br/></p></body></html>")
```

6.主程序运行:运行界面程序

参考代码：

```
if __name__ == '__main__':
    app = QApplication(argv)
    window = QMainWindow()
    ui = Ui_MainWindow(window)
    window.show()
    exit(app.exec_())
```

图 4.相机图像采集界面

六、常见问题

相机打不开，查看摄像头的 USB 线是否插紧，通过终端输入‘ls dev/*dev’指令查看是否有连接设备，程序中摄像头对应的 USB 端口是否正确。

像素尺寸测量

一、实验目的

（1）像素尺寸测量方法；

（2）掌握基于图像测量物体尺寸方法；

二、实验内容

通过测量图像的方式来测量物体尺寸，首先需要知道单个像素的尺寸，需要用固定长度的物体来做尺寸标定，比如 30mm 长的物体在图像中的像素尺寸是 212 个像素，那么每个像素尺寸为 30/212,通过图像处理的方式求得物体的像素尺寸乘以每个像素尺寸的大小就是物体的实际尺寸，此种方式需要保证物体与相机之间的距离不发生改变即物距不能发送改变，否则测量的物体尺寸是不正确的，因为物距发生改变，那么相对于的单个像素的尺寸也发生了改变。

三、实验环境

硬件环境	JetsonNX
操作系统	Linux
实验设备	USB 相机，Jetson NX
实验配件	USB 相机线

四、实验原理

1.硬件原理

通过 USB 相机采集物体图像，通过测量物体实际边长所占的像素大小即可知道该物距下单个像素的实际尺寸是多少。

2.算法原理

本实验将通过 USB 相机采集图像(视觉系统认知中有图像采集程序)，通过两点间的距离计算公式 $d=\sqrt{(x1-x2)^2+(y1-y2)^2}$ (其中(x1,y1),(x2,y2)是图像中任意两点的像素坐标)计算图像中两个点之间的像素距离。

五、实验步骤

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）打开视觉系统认知中相机图像采集程序，将教具(纯色的一面)放相机视野下白色区域中，保存一张纯颜色面图像，将保存的图像拷贝到当前程序目录下的 pic 文件夹中，并将图像的名称进行修改，同时修改图像名称后，程序中的图像名称也需要同步。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

参考代码：

```
# 导入库文件
import cv2 as cv
from math import sqrt
```

3.定义变量

参考代码：

```
global img #全局变量 img 图像
global point1, point2 #全局变量 图像中的两点
```

4.鼠标点击事件函数

参考代码：

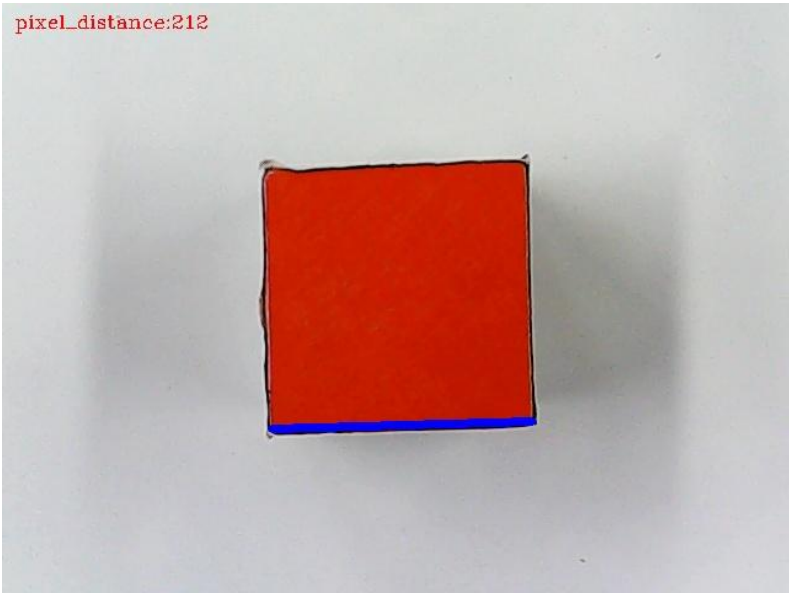
```
def on_mouse(event, x, y, flag, param):#鼠标点击事件函数
    global img, point1, point2
    img2 = img.copy()
    if event == cv.EVENT_LBUTTONDOWN:  # 左键点击
        point1 = (x, y)
        cv.circle(img2, point1, 10, (0, 255, 0), 5)#鼠标点击的位置画圈
        cv.imshow('image', img2)
    elif event == cv.EVENT_MOUSEMOVE and (flag & cv.EVENT_FLAG_LBUTTON):  # 按住左键拖曳
        cv.line(img2, point1, (x, y), (255, 0, 0), 2)#开始画线
        cv.imshow('image', img2)
    elif event == cv.EVENT_LBUTTONUP:  # 左键释放
        point2 = (x, y)
```

```
            cv.line(img2, point1, point2, (255, 0, 0), 5)
            cv.imshow('image', img2)
            pixel_distance = sqrt((point1[0] - point2[0]) ** 2 + (point1[1] - point2[1]) ** 2)#计
算两点之间的像素距离
            cv.putText(img2, "pixel_distance:" + str(round(pixel_distance)), (10, 20),
cv.FONT_HERSHEY_COMPLEX, 0.5,
                       (0, 0, 255), 1)#将距离写在图像上
            cv.imshow('image', img2)
            cv.imwrite("pic/result.jpg", img2)  # 保存结果
            print("pixel_distance:", pixel_distance)
        elif event == cv.EVENT_RBUTTONDOWN:  # 右键点击,刷新图像
            cv.imshow('image', img2)
```

4.主程序运行

参考代码：

```
if __name__ == "__main__":
    img = cv.imread('pic/1.jpg')  # 此处更改图片路径
    cv.namedWindow('image')
    cv.setMouseCallback('image', on_mouse)
    cv.imshow('image', img)
    cv.waitKey(0)
    cv.destroyAllWindows()
```



运行程序，用鼠标左键点击需要测量物体的第一个点，然后按住鼠标左键不放拖动鼠标直到第二个点位置松开鼠标左键即可测量两点之间的像素距离。如果需要重新测量，只需要松开鼠标左键，点击鼠标右键即可重新测量当前物体的像素尺寸。测量好的图像会保存在代码同级目录下的 pic 文件夹中。实验中物体的实际长度是 30mm，测量的像素尺寸是 212，单个像素的尺寸=30mm/212；

物体定位和角度测量

一、实验目的

（1）掌握图像处理过程中的方法；

（2）掌握基于图像定位物体中心并测量旋转角度；

二、实验内容

通过学习图像处理过程中的方法处理图像获取物体的中心点及角度。图像处理过程包括图像采集，图像读入，图像滤波去噪，图像预处理，图像二值化，图像形态学处理，图像轮廓获取，图像轮廓中心点及角度求取等图像处理方法。实验过程是将图像处理过程中每一个步骤进行拆分，一步一步完成图像处理得到结果。主要掌握图像处理的流程方法，此方法也是图像处理的主要流程框架，经典图像处理方式都是按照这样的基本步骤来完成，无非是中间会根据需要加判断逻辑和预处理。所以本实验主要是将图像处理的一般过程进行可视化效果呈现，调整图像处理中的参数，实现不同的效果，达到掌握图像处理一般过程方法的目的。

三、实验环境

硬件环境	JetsonNX
操作系统	Linux
实验设备	USB 相机，Jetson NX
实验配件	USB 相机线，教具

四、实验原理

1.硬件原理

通过 USB 相机采集物体图像，通过图像处理算法得到物体中心点坐标及角度测量。

2.算法原理

本实验将通过 USB 相机采集图像(视觉系统认知中有图像采集程序)，通过图像处理过程包括图像采集，图像读入，图像滤波去噪，图像预处理，图像二值化，图像形态学处理，图像轮廓获取，图像轮廓中心点及角度求取等图像处理方法，实现中点坐标求取和角度测量。

我们采集的彩色图像是一种 RGB 三通道图像。举例一张 640*480 分辨率的彩色图像，以图像的左上角为原点（0,0）,横向即 X 轴有 640 个像素单位，纵轴即 Y 轴有 480

个像素单位，每一个像素单位可以看成是一个个像元，每一个像元可以看成是一个个正方体，正方体的边长就是像元尺寸，像元尺寸就是图像传感器的最小单位尺寸，在生产图像传感器时，像元尺寸就已经确定。每一个像元就是一个像素单位，对于彩色图像，每一个像素是由三个通道组成也就是 RGB(红，绿，蓝)组成，能发出红绿蓝三种颜色的光，每一种光的亮度是 0-255,那么三通道的颜色共有 256*256*256=16777216 种颜色组合，正红色表达就是(255,0,0)，正绿色就是(0,255,0)，正蓝色(0,0,255)，纯黑色(0,0,0)，纯白色(255,255,255)。经典图像处理的过程实质上是像素的操作，也就是对每个通道进行像素分析，从而判断是需要还是不需要。那么这个分析的过程就是一个个图像处理的算法，很多个这样的处理处理分析算法组成就可以完成图像分析检测识别等。

五、实验步骤

图像滤波

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）打开视觉系统认知中相机图像采集程序，将教具(纯色的一面放相机视野下白色区域中，保存一张纯颜色面图像，将保存的图像拷贝到当前程序目录下的 pic 文件夹中，并将图像的名称进行修改，同时修改图像名称后，程序中的图像名称也需要同步。

（5）物体定位和角度测量一共分为 5 个大步骤，分为 1.图像滤波,2.图像预处理，3.图像二值化，4.图像形态学处理，5.图像轮廓获取及求取物体坐标和角度。从图像滤波开始，首先读取图像采集程序采集的图像，经过滤波操作会在 pic 文件夹保留滤波后的效果图，图像预处理的输入图片就是上一步滤波后保留的图像，依次类推，每一个步骤的读入图像都是上一步骤的输出图像。按照此过程操作，可以详细的查看每一个操作步骤的图像处理结果，更加直观的了解图像处理过程。

注意：在 jupyter lab 新建的文件载入 python 的 py 程序文件，只需要在新建的 jupyter lab 程序文件输入%load 程序名称.py 点击运行按钮即可将 py 文件载入到 jupyter lab 程序文件中(例：%load 1.滤波.py)，两个文件需要在同一个文件夹。

如果需要在多个单元格载入多个 py 文件，需要将 key = cv2.waitKey(0)(窗口暂停)cv2.destroyALLWindows()(窗口销毁)两行代码注释掉（注释快捷键 Ctrl+/）只保留最后一个窗口暂停和销毁窗口的两行代码。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

参考代码：

```
# 导入库文件
import cv2
```

3.读取采集程序采集的图像

参考代码：

```
image = cv2.imread("pic/1.jpg",-1)#读取图片
    # image = cv2.imread(“名字”, 格式)
    # 其中 image 是读入的图像名称，“名字”指的是需要读取的图片的文件路径及名字。
    # 格式指的是读取的标记————————-1 表示保持原有的格式不变
    # 0 表示将图像调整为单通道的灰度图像
    # 1 表示将图像调整为 3 通道的 BGR 通道。为默认值
    # 2 表示的是当载入的图像额外 16 位或者 32 位时，就返回其对应的深度图像；否则，将转换为 8 为图像
    # 4 表示的是以任何可能的颜色格式读取图像
```

4. 滤波

高斯滤波，滤波的作用是使图像更加的平滑，去除图像噪点的作用，滤波的方法有很多，常用的有高斯滤波，中值滤波，均值滤波，方框滤波等。每一种滤波都有相对的优点和缺点，根据自己的图像处理需求选择合适的滤波方式，高斯滤波的特点是边界保留效果更好，我们的图像处理目的是获取物体的中心点坐标和角度，那么首先就需要找到物体的轮廓才能对中心点坐标进行计算，所以选择高斯滤波，这里可以尝试使用中值

滤波，均值滤波看看效果如何。

滤波的卷积核参数（25,25）可以根据图像效果调节大小，值越大，图像越模糊。

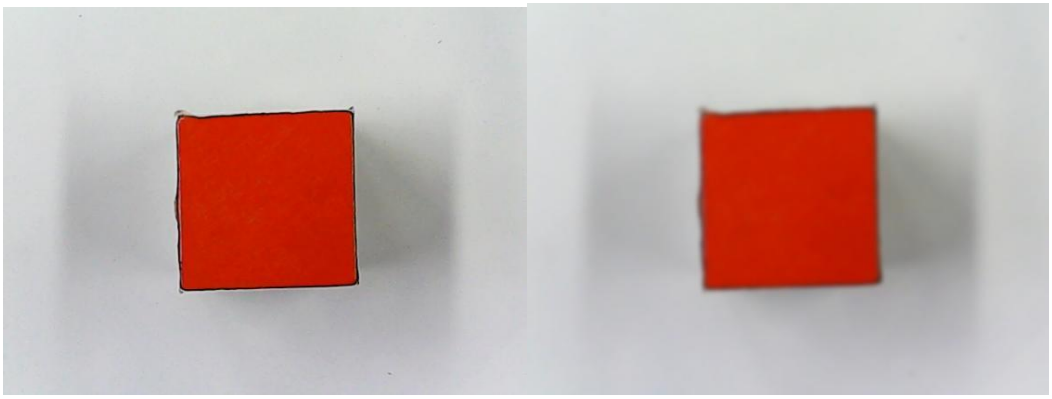
参考代码：

```
gauss = cv2.GaussianBlur(image, (25, 25), 0)#高斯滤波
```

4.保存图像，显示窗口

参考代码：

```
cv2.imshow("gauss", gauss)#显示图像
cv2.imwrite('pic/gauss.jpg', gauss)#保存图像
key = cv2.waitKey(0)#窗口暂停
cv2.destroyAllWindows()
```



原图 （1.jpg）　　高斯滤波后图像（gauss.jpg）

图像预处理

打开 jupyter lab，可以基于图像滤波同一个 ipynb 文件，也可以另外新建程序文件，基于同一个 ipynb 文件新建的单元格运行时，需要结束当前程序运行的内核，将程序行选择到当前程序块，点击运行即可。

图像预处理是可以解释为使图像达到理想的处理效果前的一些处理步骤。以该实验为例，实验目的是对不同形状，不同颜色的色块进行定位和角度测量。那么就需要识别物体的轮廓，为了使物体的轮廓提取时明显，且增加提取的稳定性，必须要对图像进行处理，如果处理的方式仅适用于一张图像，换了其他图像和不同情况下拍摄的有差异图像，算法就不能适应，这样算法的适应性就不强，那么为了达到算法适应性强的目的，就需要前期对图像进行很多的预处理，尽量减少因为图像差异造成的不稳定性。

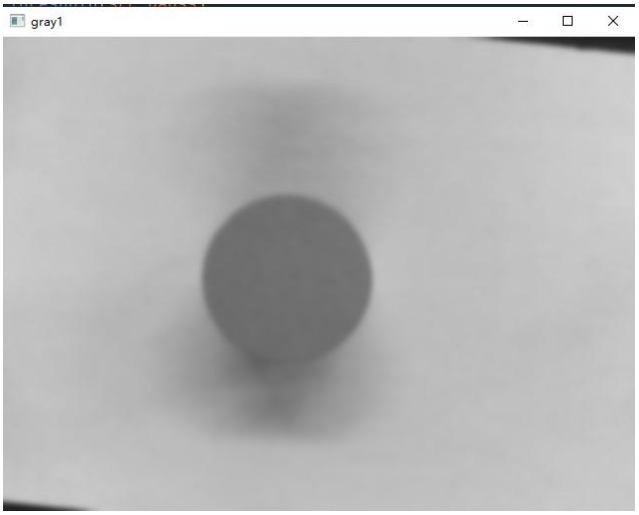
那么现在讲一下图像预处理的思路，我们常规的图像处理流程首先就是对彩色图像灰度化,灰度化就是像素值归一处理，就是让三通道的图像变成单通道图像，单通道图像每个像素点用一个值表示颜色，它的像素值在 0 到 255 之间，0 是黑色，255 是白色，中间值是一些不同等级的灰色，为什么需要转成单通道进行处理呢？因为后面处理图像是一个二值图像，二值图像是一副 0 和 1 组成的图像，也就是黑白图像，我们可以根据

黑白在图像中对物体进行分块，再根据一些块的面积，形状，尺寸等特征信息进行筛选我们要的物体块。二值图像是由灰度图像而来，灰度图像是单通道图像，我们可以根据像素值的大小对像素进行筛选，保留有用的像素。

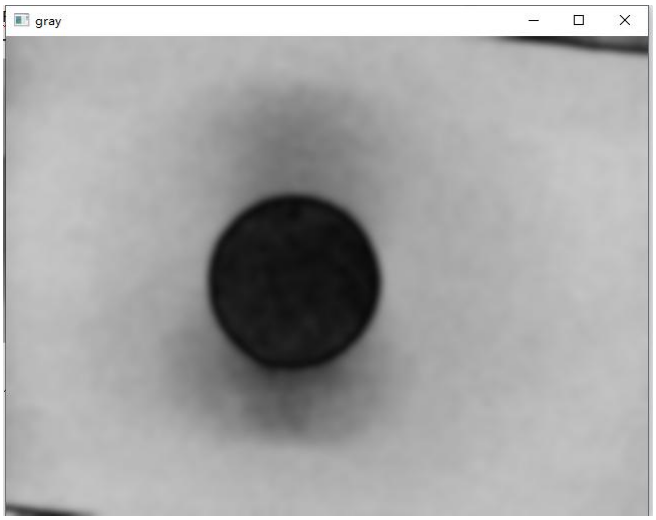
获取单通道图像我们可以通过 opencv 中 cvtColor 函数直接将 RGB 图像转成灰度，也可以通过 split 直接对 RGB 图像进行通道分离，变成 R,G,B 三个单通道的图像。我们常规的做法一般都是直接转灰度，但是为了保证不同物体间的图像差别，需要进行一些增强的操作。因为我们需要识别不同形状和颜色的物体，经过测试，将 RGB 转成 HSV 再进行通道分割后做差分处理得到的图像的稳定性符合要求。

1.直接将 RGB 图像转成灰度图效果

```
gray1=cv2.cvtColor(src_gauss,cv2.COLOR_BGR2GRAY)
```



2.将 RGB 转 HSV 后三通道分割做差分取像素值最大的灰度图效果如下



我们可以很直观的看到，第二种方式处理的图像，得到的图像对比度比第一张的图像强很多，这样我们在做二值化的时候就可以很精准的提取到物体的轮廓，以上就是图像做预处理的目的和意义，就是增加前景图和背景图的对比度为后续处理提供一个

可靠的图像。

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）图像预处理实验是基于图像滤波实验的图像结果来做的。

（5）新建单元格输入“%load 2.图像预处理.py”点击运行载入滤波后图像。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

参考代码：

```
# 导入库文件
import cv2
import numpy as np
```

3.图像预处理函数

参考代码：

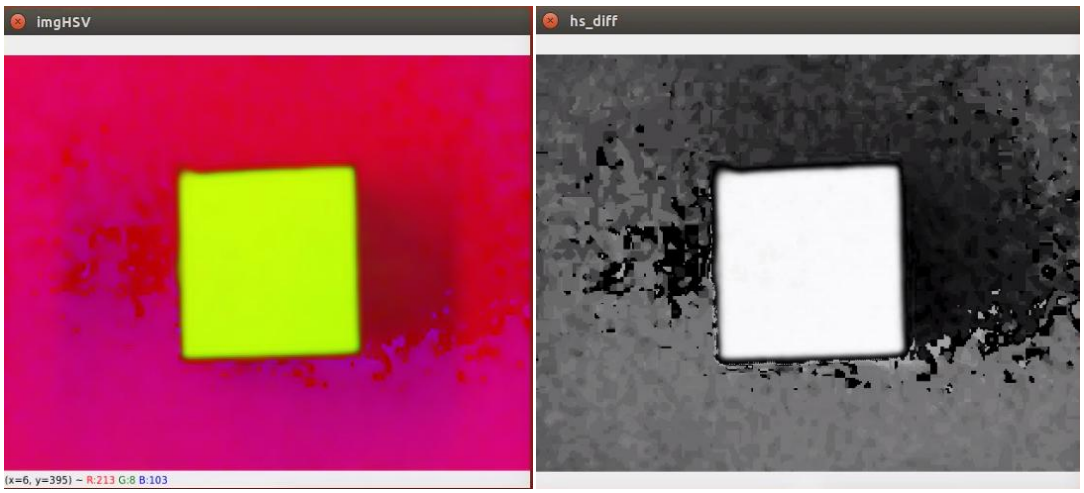
```
# 图像预处理
def img_Threshold(src_gauss):
    imgHSV = cv2.cvtColor(src_gauss, cv2.COLOR_BGR2HSV)
    # gray1=cv2.cvtColor(src_gauss, cv2.COLOR_BGR2GRAY)
    # cv2.imshow("gray1", gray1)
    cv2.imshow("imgHSV", imgHSV)
    # # 三通道分割
    h, s, v = cv2.split(imgHSV)
    # 检测差值
    hs_diff = cv2.absdiff(h, s)
    cv2.imshow("hs_diff", hs_diff)
    sv_diff = cv2.absdiff(s, v)
    cv2.imshow("sv_diff", sv_diff)
    hv_diff = cv2.absdiff(h, v)
    cv2.imshow("hv_diff", hv_diff)
    v1 = np.mean(hs_diff)  # 取每个通道的均值
    v2 = np.mean(sv_diff)
    v3 = np.mean(hv_diff)
    v_max = (v1 if v1 > v2 else v2) if (v1 if v1 > v2 else v2) > v3 else v3  # 比较均值得到最大值
```

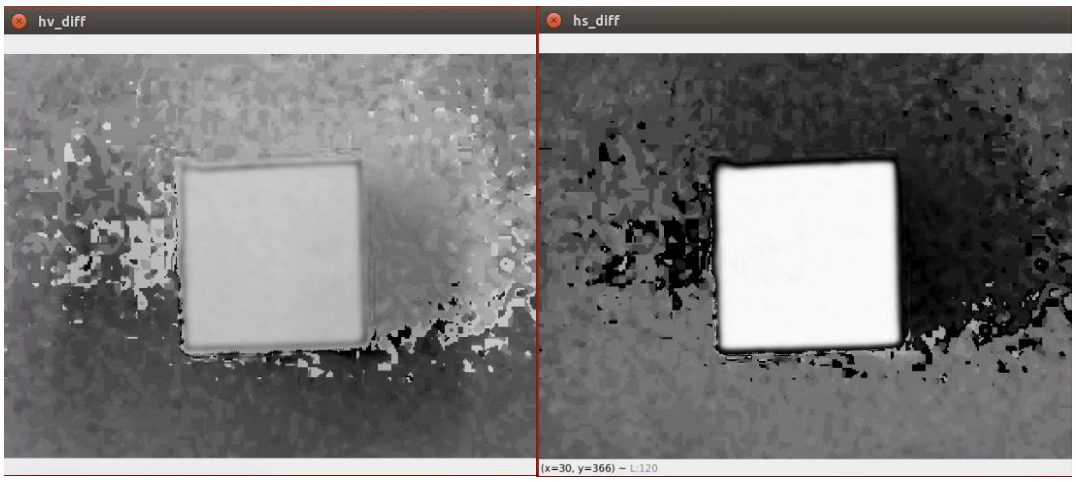
```
    print(v_max, v3)
    if v_max > 8:
        if abs(v_max - v1) < 0.01:
            gray = hs_diff.copy()
        elif abs(v_max - v2) < 0.01:
            gray = sv_diff.copy()
        elif abs(v_max - v3) < 0.01:
            gray = hv_diff.copy()
    return gray
```

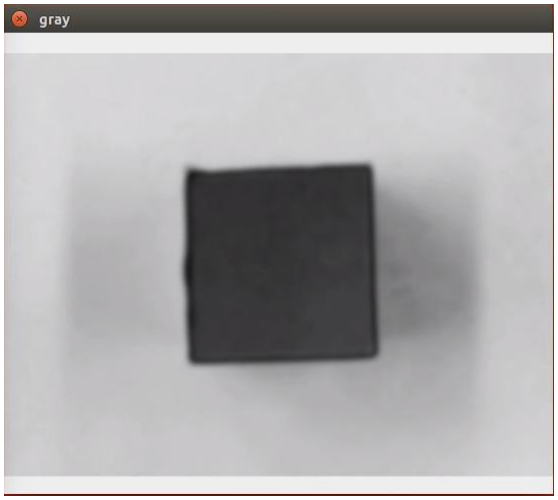
4.主程序运行，显示处理图像

参考代码：

```
if __name__ == "__main__":
    gauss = cv2.imread("pic/gauss.jpg", -1)
    # 图像预处理
    gray = img_Threshold(gauss)
    # 高斯滤波
    gray = cv2.GaussianBlur(gray, (15, 15), 3)
    cv2.imshow("gray", gray)  # 显示图像
    cv2.imwrite('pic/gray.jpg', gray)  # 保存图像
    key = cv2.waitKey(0)  # 窗口暂停
    cv2.destroyAllWindows()
```







通过图像预处理将高斯滤波后的图像转成 HSV 进行三通道分割，将分割后的单通道图像进行图像做差，通过判断做差后的图像均值，选择对比度最大的图像作为最后预处理的灰度图像。

图像二值化

图像二值化前面提到是将灰度图像二值化变成黑白图像，二值化图像常用的方法是根据设定的阈值，遍历像素判断灰度图像每个像素值与设定的阈值的大小，比阈值大就将该像素值设为 0，比阈值小就设置为 255，或者相反。这样就将灰度图变成了黑白图像。Opencv 中二值化函数有简单阈值 cv2. threshold，自适应阈值 cv2. adaptiveThreshold。

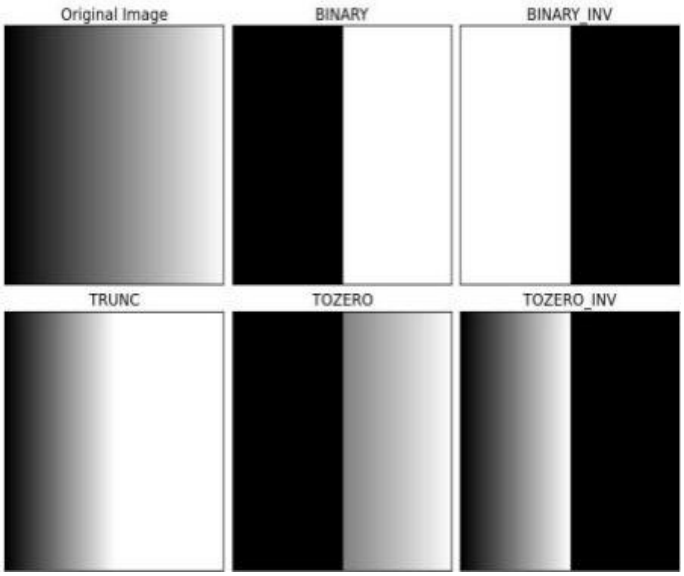
1. 简单阈值

与名字一样，这种方法非常简单。但像素值高于阈值时，我们给这个像素

赋予一个新值（可能是白色），否则我们给它赋予另外一种颜色（也许是黑色）。

这个函数就是 cv2. threshhold()。这个函数的第一个参数就是原图像，原图像应该是灰度图。第二个参数就是用来对像素值进行分类的阈值。第三个参数就是当像素值高于（有时是小于）阈值时应该被赋予的新的像素值。OpenCV 提供了多种不同的阈值方法，这是有第四个参数来决定的。这些方法包括：

- cv2.THRESH_BINARY
- cv2.THRESH_BINARY_INV
- cv2.THRESH_TRUNC
- cv2.THRESH_TOZERO
- cv2.THRESH_TOZERO_INV



2. 自适应阈值

在前面的部分我们使用是全局阈值，整幅图像采用同一个数作为阈值。当时这种方法并不适应与所有情况，尤其是当同一幅图像上的不同部分的具有不同亮度时。这种情况下我们需要采用自适应阈值。此时的阈值是根据图像上的每一个小区域计算与其对应的阈值。因此在同一幅图像上的不同区域采用的是不同的阈值，从而使我们能在亮度不同的情况下得到更好的结果。这种方法需要我们指定三个参数，返回值只有一个。

- Adaptive Method- 指定计算阈值的方法。
  - cv2. ADPTIVE_THRESH_MEAN_C：阈值取自相邻区域的平均值
  - cv2. ADPTIVE_THRESH_GAUSSIAN_C：阈值取值相邻区域的加权和，权重为一个高斯窗口。
- Block Size - 邻域大小（用来计算阈值的区域大小）。
- C - 这就是是一个常数，阈值就等于的平均值或者加权平均值减去这个常数。

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）图像预处理实验是基于图像预处理实验的图像结果来做的。

（5）新建单元格输入“%load 3.图像二值化.py”点击运行载入滤波后图像。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

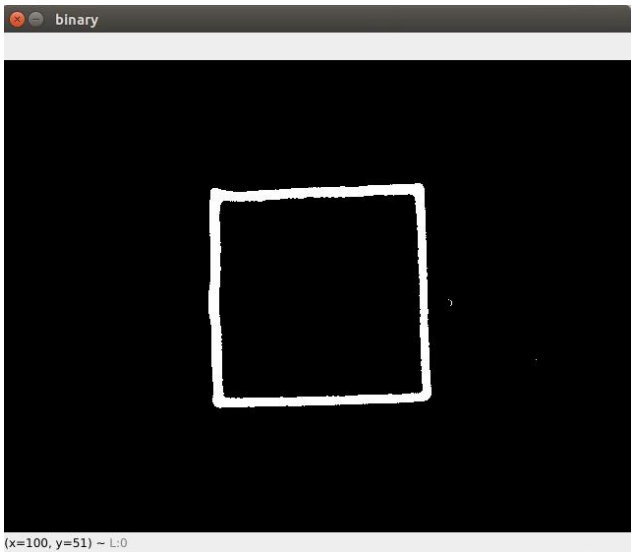
参考代码：

```
# 导入库文件
import cv2
```

3.主程序运行，显示阈值图像

参考代码：

```
if __name__ == "__main__":
    gray = cv2.imread("pic/gray.jpg", -1)
    # 高斯滤波
    gray = cv2.GaussianBlur(gray, (25, 25), 3)
    # 自适应阈值二值化
    binary = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY_INV, 11, 2)
    cv2.imshow("binary", binary)  # 显示图像
    cv2.imwrite('pic/binary.jpg', binary)  # 保存图像
    key = cv2.waitKey(0)  # 窗口暂停
    cv2.destroyAllWindows()
```



注意：如果图像中的噪点较多，可以通过调节滤波的参数来去掉噪点。

图像形态学处理

形态学操作是根据图像形状进行的简单操作。一般情况下对二值化图像进行的操作。需要输入两个参数，一个是二值图像，第二个被称为结构化元素或核，它是用来决定操作的性质的。两个基本的形态学操作是腐蚀和膨胀。他们的变体构成了开运算，闭运算，梯度等。

形态学操作有腐蚀，膨胀，开运算，闭运算等，函数有：cv2. erode()，cv2. dilate()，cv2. morphologyEx()

等。

1. 腐蚀(cv2. erode)

就像土壤侵蚀一样，这个操作会把前景物体的边界腐蚀掉（但是前景仍然是白色）。这是怎么做到的呢？卷积核沿着图像滑动，如果与卷积核对应的原图像的所有像素值都是 1，那么中心元素就保持原来的像素值，否则就变为零。这回产生什么影响呢？根据卷积核的大小靠近前景的所有像素都会被腐蚀掉（变为 0），所以前景物体会变小，整幅图像的白色区域会减少。这对于去除白噪声很有用，也可以用来断开两个连在一块的物体等。

举例：

```
import cv2
import numpy as np
img = cv2. imread('j. png', 0)
kernel = np. ones((5, 5), np. uint8)
erosion = cv2. erode(img, kernel, iterations = 1)
```

结果：

2. 膨胀(cv2. dilate)

与腐蚀相反，与卷积核对应的原图像的像素值中只要有一个是 1，中心元素的像素值就是 1。所以这个操作会增加图像中的白色区域（前景）。一般在去噪声时先用腐蚀再用膨胀。因为腐蚀在去掉白噪声的同时，也会使前景对象变小。所以我们再对他进行膨胀。这时噪声已经被去除了，不会再回来了，但是前景还在并会增加。膨胀也可以用来连接两个分开的物体。

举例：

dilation = cv2. dilate(img, kernel, iterations = 1)

结果：

3. 开运算(cv2. morphologyEx)

先进性腐蚀再进行膨胀就叫做开运算。就像我们上面介绍的那样，它被用来去除噪声。

举例：

opening = cv2. morphologyEx(img, cv2. MORPH_OPEN, kernel)

结果：

4. 闭运算(cv2. morphologyEx)

先膨胀再腐蚀。它经常被用来填充前景物体中的小洞，或者前景物体上的小黑点。

举例：

closing = cv2. morphologyEx(img, cv2. MORPH_CLOSE, kernel)

结果：

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）图像形态学处理实验是基于图像二值化实验的图像结果来做的。

（5）新建单元格输入“%load 4.图像形态学处理.py”点击运行载入滤波后图像。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

参考代码：

```
# 导入库文件
import cv2
```

3.形态学处理函数

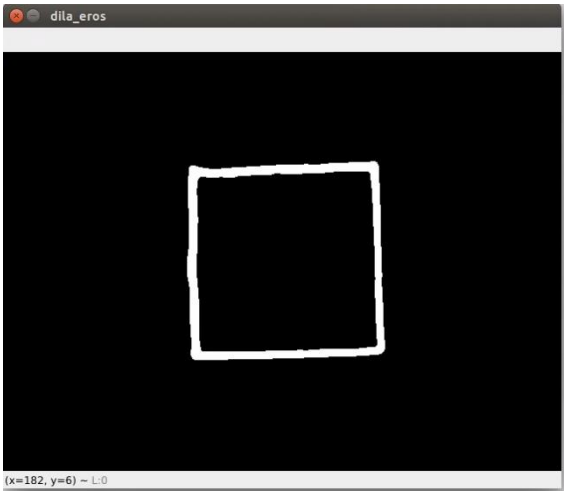
参考代码：

```
# 图像开运算和闭运算(形态学处理)
def img_dila_eros(src_img):
    # 3. 膨胀和腐蚀操作的核函数
    element1 = cv2.getStructuringElement(cv2.MORPH_RECT, (4, 3))
    element2 = cv2.getStructuringElement(cv2.MORPH_RECT, (4, 3))
    element3 = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
    # 4. 膨胀一次，让轮廓突出
    src_img = cv2.dilate(src_img, element2)
    # 5. 腐蚀一次，去掉细节，如表格线等。注意这里去掉的是竖直的线
    src_img = cv2.erode(src_img, element1)
    # 6. 再次膨胀，让轮廓明显一些
    # src_img = cv2.dilate(src_img, element2)
    src_img = cv2.morphologyEx(src_img, cv2.MORPH_OPEN, element3)#开运算去掉噪点
    return src_img
```

4.主程序运行，显示形态学处理图像

参考代码：

```
if __name__ == "__main__":
    binary = cv2.imread("pic/binary.jpg", -1)
    # 图像形态学处理
    dila_eros = img_dila_eros(binary)
    cv2.imshow("dila_eros", dila_eros)
    cv2.imwrite('pic/dila_eros.jpg', dila_eros)  # 保存图像
    key = cv2.waitKey(0)  # 窗口暂停
    cv2.destroyAllWindows()
```



寻找轮廓（定位和角度计算）

轮廓可以简单认为成将连续的点（连着边界）连在一起的曲线，具有相同的颜色或者灰度。轮廓在形状分析和物体的检测和识别中很有用。

• 为了更加准确，要使用二值化图像。在寻找轮廓之前，要进行阈值化处理或者 Canny 边界检测。

• 查找轮廓的函数会修改原始图像。如果你在找到轮廓之后还想使用原始图像的话，你应该将原始图像存储到其他变量中。

• 在 OpenCV 中，查找轮廓就像在黑色背景中超白色物体。你应该记住，要找的物体应该是白色而背景应该是黑色。

寻找轮廓前面提及到是对二值图像进行处理，把二值图像中的黑白按照轮廓提取出来，根据轮廓的面积，尺寸等因素来获取我们想要的物体轮廓。获取物体的轮廓后就可以对轮廓进行处理，比如轮廓多边形拟合，求取轮廓的最小外接矩形(斜矩形)，求取轮廓的角度等。主要会用到以下几个函数：

1. 找轮廓（cv2. findContours）

函数 cv2. findContours() 有三个参数，第一个是输入图像，第二个是轮廓检索模式，第三个是轮廓近似方法。返回值有三个，第一个是图像，第二个是轮廓，第三个是（轮廓的）层析结构。轮廓（第二个返回值）是一个 Python 列表，其中存储这图像中的所有轮廓。每一个轮廓都是一个 Numpy 数组，包含对象边界点（x，y）的坐标。

上边我们已经提到轮廓是一个形状具有相同灰度值的边界。它会存贮形状边界上所有的（x，y 坐标。但是需要将所有的这些边界点都存储吗？这就是这个参数要告诉函数 cv2. findContours 的。这个参数如果被设置为 cv2. CHAIN_APPROX_NONE，所有的边界点都会被存储。但是我们真的需要这么多点吗？例如，当我们找的边界是一条直线时。你用需要直线上所有的点来表示直线吗？不是的，我们只需要这条直线的两个端点而已。这就是 cv2. CHAIN_APPROX_SIMPLE 要做的。它会将轮廓上的冗余点都去

掉，压缩轮廓，从而节省内存开支。我们用下图中的矩形来演示这个技术。在轮廓列表中的每一个坐标上画一个蓝色圆圈。第一个图显示使用 cv2. CHAIN_APPROX_NONE 的效果，一共 734 个点。第二个图是使用 cv2. CHAIN_APPROX_SIMPLE 的结果，只有 4 个点。看到他的威力了吧！

3. 绘制轮廓（cv2. drawContours()）

函数 cv2. drawContours() 可以被用来绘制轮廓。它可以根据你提供的边界点绘制任何形状。它的第一个参数是原始图像，第二个参数是轮廓，一个 Python 列表。第三个参数是轮廓的索引（在绘制独立轮廓是很有用，当设置为 -1 时绘制所有轮廓）。

4. 轮廓面积（cv2. contourArea()）

轮廓的面积可以使用函数 cv2. contourArea() 计算得到。

实验中通过打印输出轮廓的面积来根据轮廓的面积筛选我们需要的轮廓，如下。

```
# 循环轮廓，判断每一个形状
for cnt in contours:
    # 获取轮廓面积
    area = cv2.contourArea(cnt)
    print("轮廓像素面积:", area)  # 打印所有轮廓面积
    # 当面积大于 20000，代表有形状存在
    if area > 20000:
        # print("轮廓像素面积:", area)  # 打印符合条件轮廓面积
```

```
轮廓像素面积: 0.0
轮廓像素面积: 0.0
轮廓像素面积: 0.0
轮廓像素面积: 42.5
轮廓像素面积: 139.5
轮廓像素面积: 0.0
轮廓像素面积: 0.0
轮廓像素面积: 0.0
轮廓像素面积: 0.5
轮廓像素面积: 2.0
轮廓像素面积: 0.0
轮廓像素面积: 0.0
轮廓像素面积: 0.0
轮廓像素面积: 0.0
轮廓像素面积: 0.0
轮廓像素面积: 50136.0
approx: 4
```

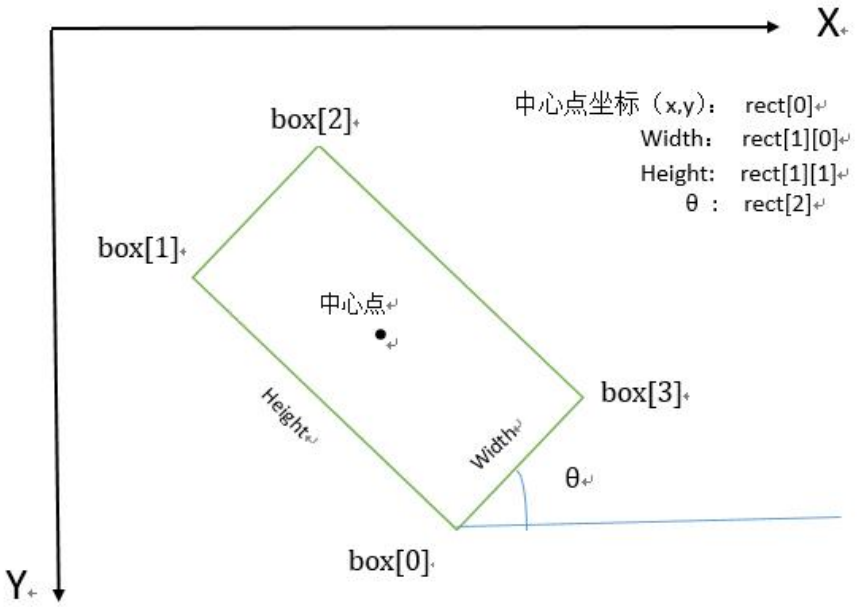
通过输出找到的轮廓面积，可以知道，50136. 0 是物体的轮廓像素面积，其余的都是噪点轮廓面积，所以我们就可以把参数设置为 20000, 滤掉噪点轮廓获取我们需要的轮廓，判断轮廓的面积参数需要自己根据打印的值进行修改。

5. 多边形拟合（cv2. approxPolyDP()）

输出近似图像折点坐标 主要功能是把一个连续光滑曲线折线化，用以判断物体是几边形。

6. 最小外接斜矩形（cv2. minAreaRect()）

用到的函数为 cv2. minAreaRect()。返回的是一个 Box2D 结构，其中包含矩形左上角角点的坐标（x，y），矩形的宽和高（w，h），以及旋转角度。但是要绘制这个矩形需要矩形的 4 个角点，可以通过函数 cv2. boxPoints() 获得。



1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）寻找轮廓实验是基于图像形态学处理实验的图像结果来做的。

（5）新建单元格输入“%load 5.寻找轮廓.py”点击运行载入滤波后图像。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

参考代码：

```
# 导入库文件
import cv2
import numpy as np
```

3.定位和角度测量函数

参考代码：

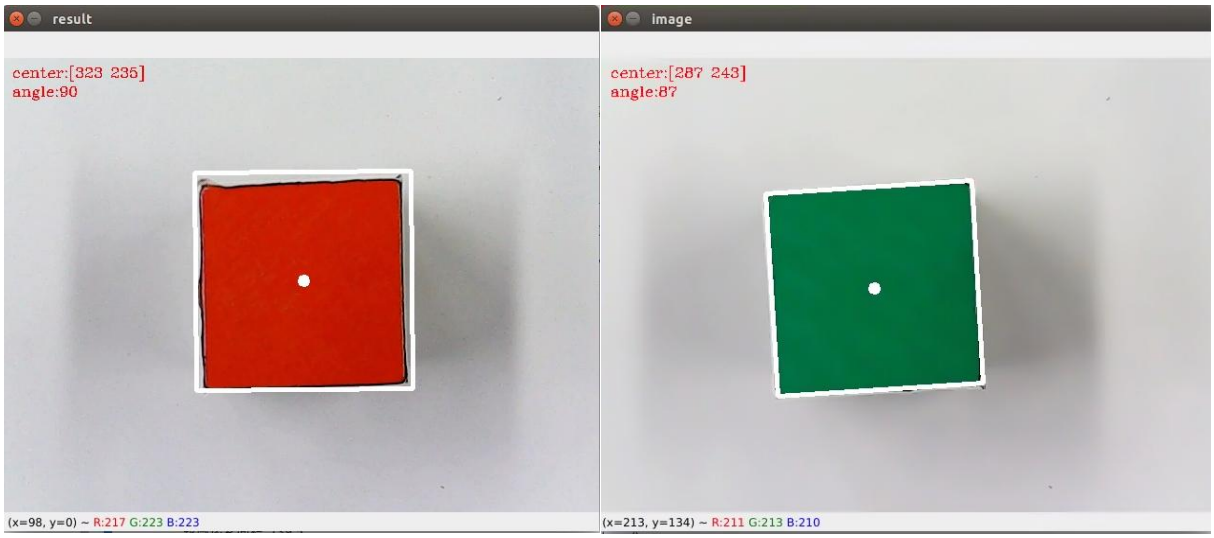
```
# 定位和角度测量
def getContours(src, img):
    # 查找轮廓，cv2.RETR_ExTERNAL=获取外部轮廓点，CHAIN_APPROX_NONE = 得到所有的像素点,CHAIN_APPROX_SIMPLE=得到轮廓的四个点
    contours, hierarchy = cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    # 循环轮廓，判断每一个形状
    for cnt in contours:
        # 获取轮廓面积
        area = cv2.contourArea(cnt)
        print("轮廓像素面积:", area)  # 打印所有轮廓面积
        # 当面积大于 20000，代表有形状存在
        if area > 20000:
            # print("轮廓像素面积:", area)  # 打印符合条件轮廓面积
            # 计算所有轮廓的周长，便于做多边形拟合
            # 多边形拟合，获取每个形状的边
            approx = cv2.approxPolyDP(cnt, 0.02 * cv2.arcLength(cnt, True), True)  # 拟合的多边形的边数
            print("approx:", len(approx))
            objCor = len(approx)  # 轮廓的边长
            rect = cv2.minAreaRect(approx)  # 最小外接矩形
            box = cv2.boxPoints(rect)  # boxPoints 返回四个点顺序：右下→左下→左上→右上
            box = np.int0(box)
            center = rect[0]  # 中心坐标
            center_array = np.array(center)
            int_center = center_array.astype(int)
            angle = rect[2]  # 旋转角度
            # 画出边界
            if objCor > 4:
                cv2.circle(src, (int(rect[0][0]), int(rect[0][1])), int(rect[1][0] / 2), (255, 255, 255), 5)
            else:
                cv2.drawContours(src, [box], 0, (255, 255, 255), 3)  # 画出多边形形状
            cv2.circle(src, (int(rect[0][0]), int(rect[0][1])), 3, (255, 255, 255), 5)
            # 画中心，写角度
            cv2.putText(src, "center:" + str(int_center), (10, 20),
```

```
cv2. FONT_HERSHEY_COMPLEX, 0. 5, (0, 0, 255), 1)
                if objCor <= 4:
                    cv2. putText(src, "angle:" + str(round(angle)), (10, 40),
cv2. FONT_HERSHEY_COMPLEX, 0. 5, (0, 0, 255), 1)
                else:
                    cv2. putText(src, "angle:" + "0", (10, 40), cv2. FONT_HERSHEY_COMPLEX,
0. 5, (0, 0, 255), 1)
```

4.主程序运行，显示定位角度处理图像

参考代码：

```
if __name__ == "__main__":
    dila_eros = cv2. imread("pic/dila_eros. jpg", -1)
    result = cv2. imread("pic/1. jpg", -1)  # 读取原图
    # 定位和角度测量
    getContours(result, dila_eros)
    cv2. imshow("result", result)
    cv2. imwrite('pic/result. jpg', result)  # 保存图像
    key = cv2. waitKey(0)  # 窗口暂停
    cv2. destroyAllWindows()
```



1 智能传感系统认知

一、实验目的

（1）了解智能传感系统的硬件布局。

（2）熟悉智能传感系统的硬件系统结构；

（3）了解智能传感系统的硬件系统原理；

（4）熟悉智能传感系统的端口资源分配；

二、实验内容

本实验要求在熟悉智能传感系统硬件结构的基础上，掌握系统的硬件系统原理和端口资源分配，通过运行演示程序，了解系统的功能。

三、实验环境

硬件环境	JETSON XAVIET NX 计算力 21TOPS, 内存 8GB
操作系统	Linux ARM64
编译环境	Arduino IDE
实验设备	人工智能实验箱智能传感系统
实验配件	键盘、鼠标、电源 12V/4A

四、实验原理

1. 智能传感系统认知

智能传感系统硬件布局如图 1 所示，其包含部件如下：

（1） 采用 8 位高性能 ATMEGA2560 处理器，256Kb Flash；

（2） 数码管显示；

（3） OLED 液晶显示屏；

（4） 温湿度传感器；

（5） 气压传感器；

（6） 亮度传感器；

（7） 超声波传感器；

（8） 火焰传感器；

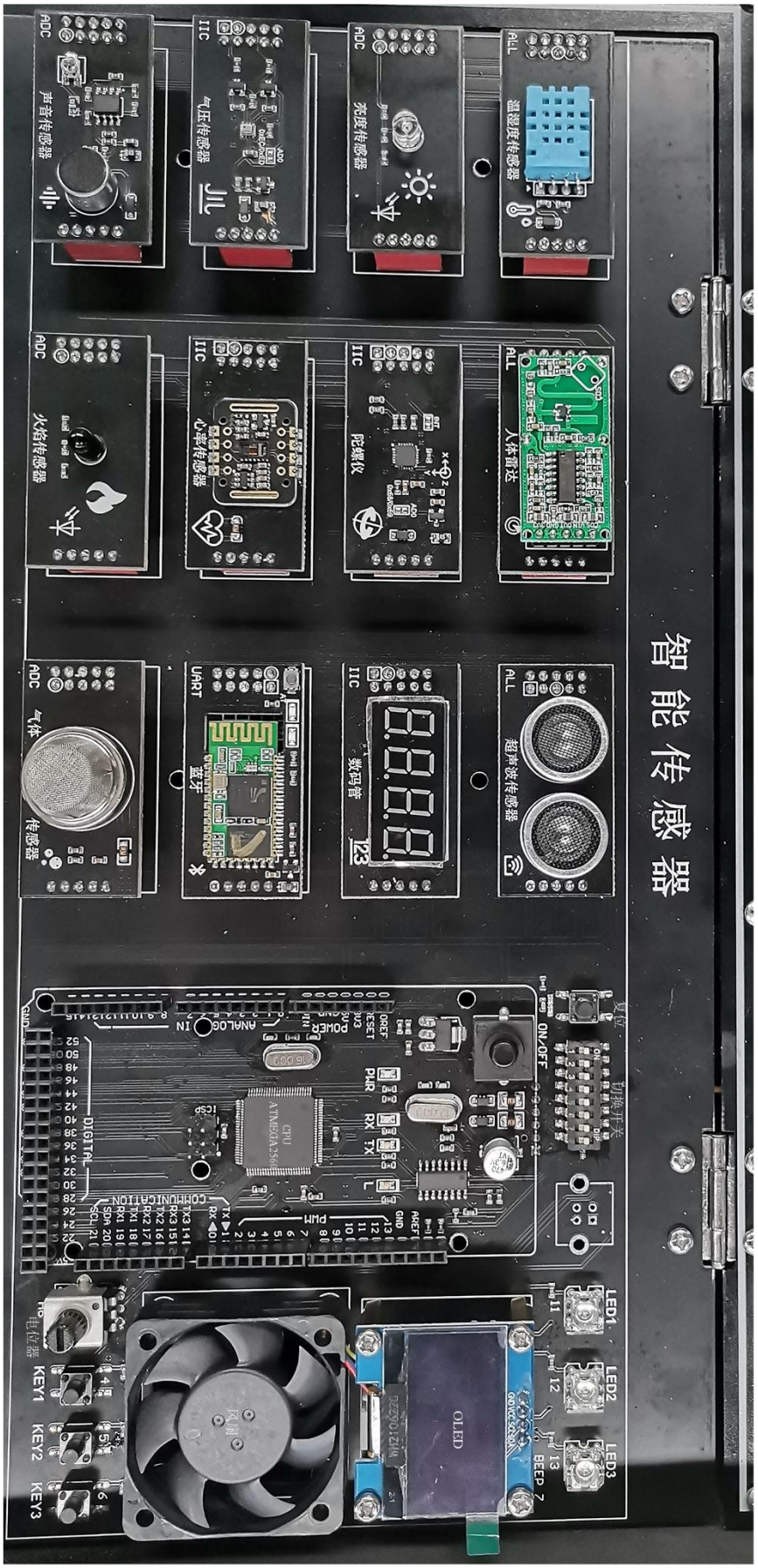
（9） 气体传感器；

（10） 心率传感器；

（11） 人体雷达；

（12） 蓝牙通讯；

（13） 云台舵机控制；


智能传感器
温湿度传感器
亮度传感器
气压传感器
声音传感器
人体雷达
陀螺仪
心率传感器
火焰传感器
超声波传感器
数码管
蓝牙
气体
传感器
ON/OFF
OLED
KEY1
KEY2
KEY3
LED1
LED2
LED3
BEEP

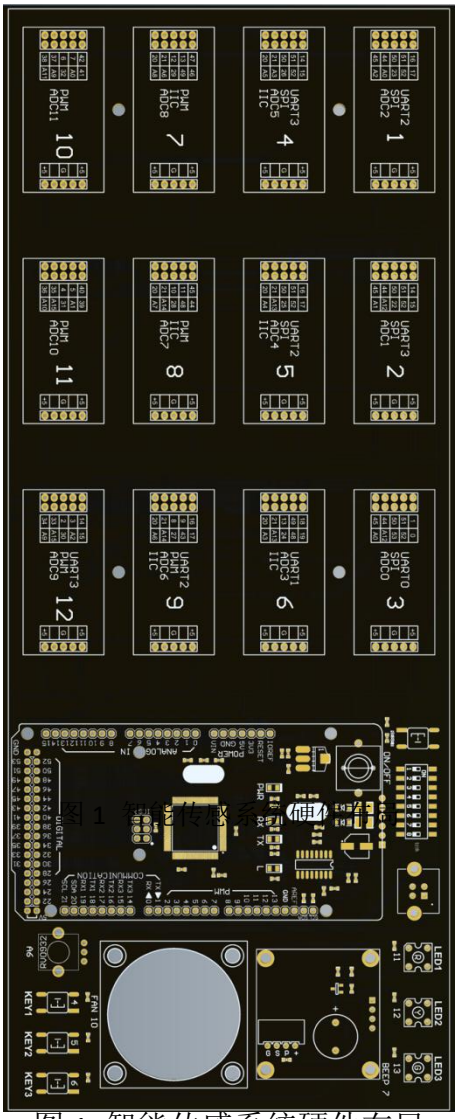


图 1 智能传感系统硬件布局

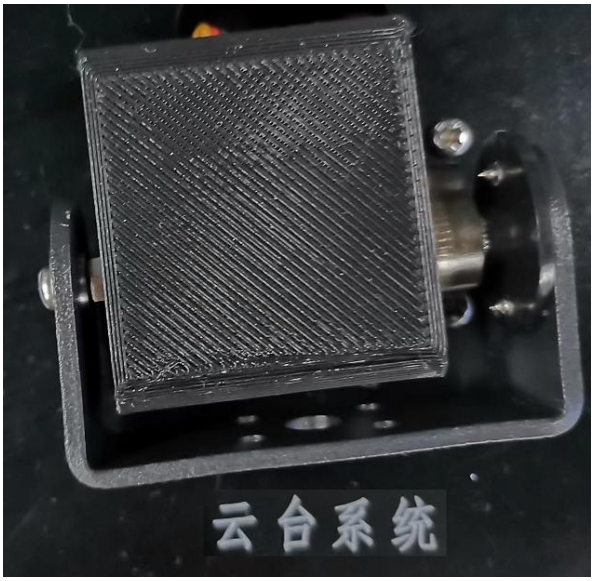


图 2 云台

（14） 陀螺仪；

（15） 蜂鸣器；

（16） 3 个 LED 灯；

（17） 4 个按钮；

（18） 风扇。

2. 智能传感系统硬件原理框图

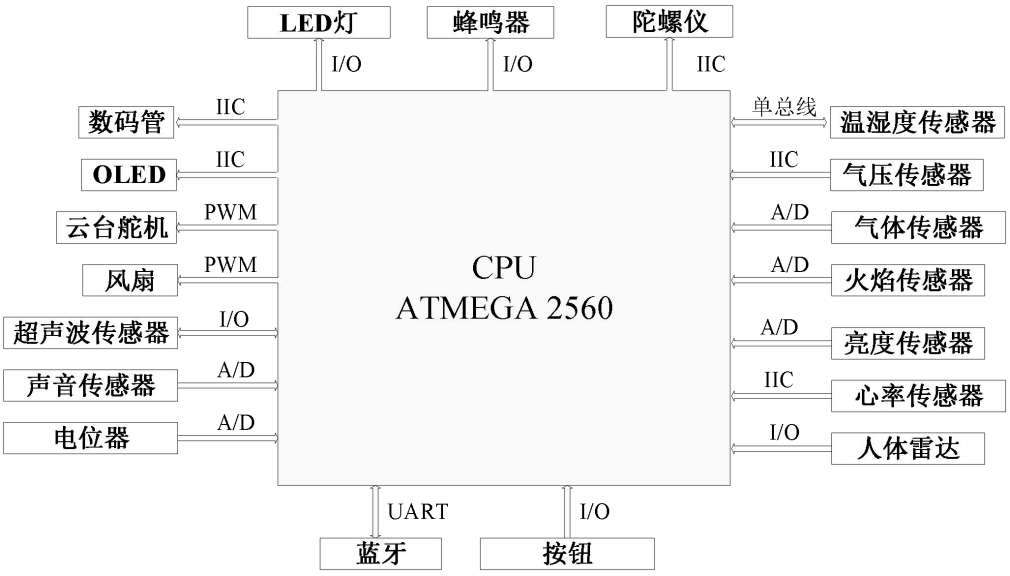


图 3 智能传感系统硬件原理框图

3. 智能传感系统硬件原理图

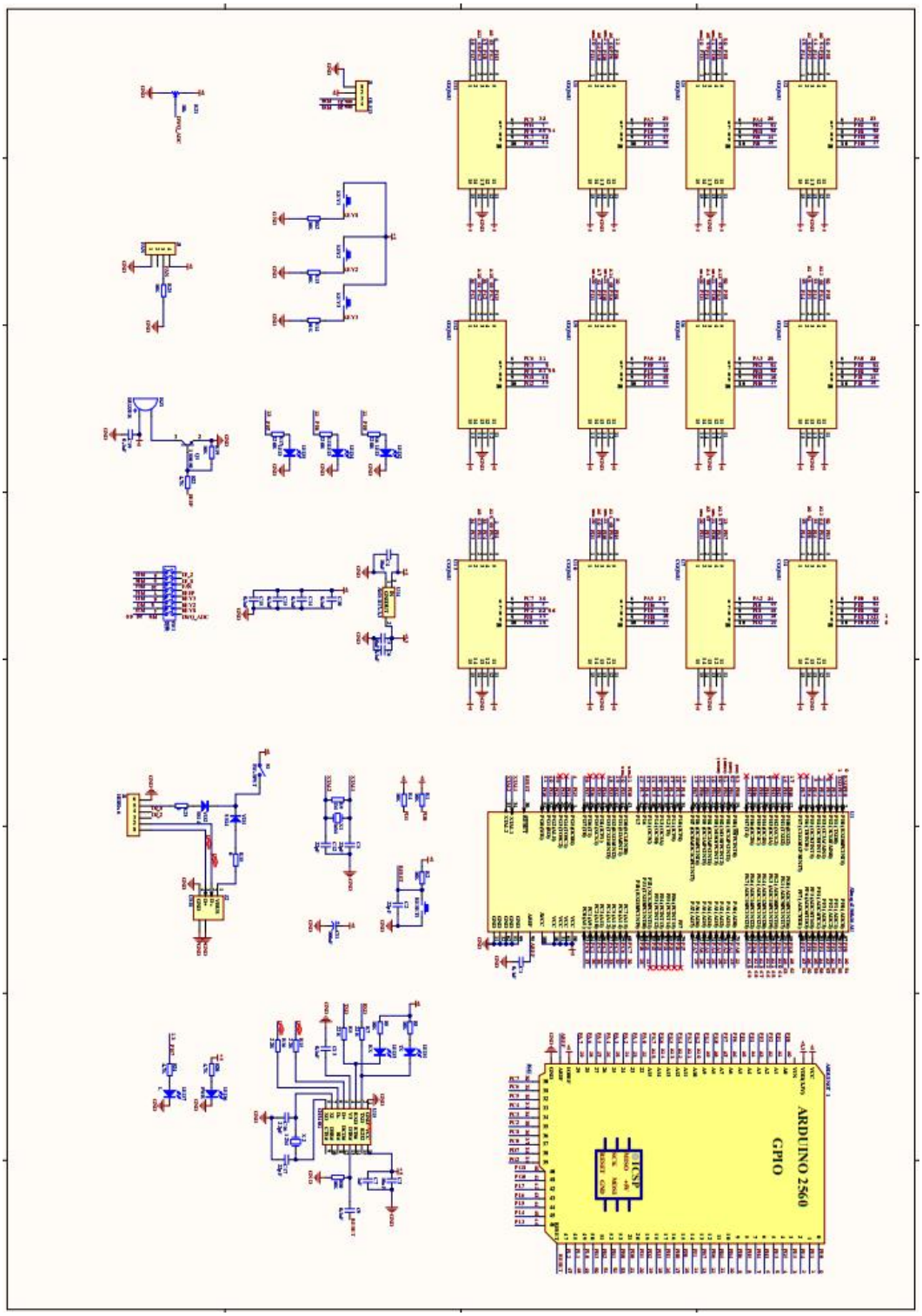


图 4 智能传感系统硬件原理图

4. 智能传感系统端口资源分配

（1）CPU ATMEGA2560 引脚图

CPU ATMEGA2560 引脚图如图 5 所示。

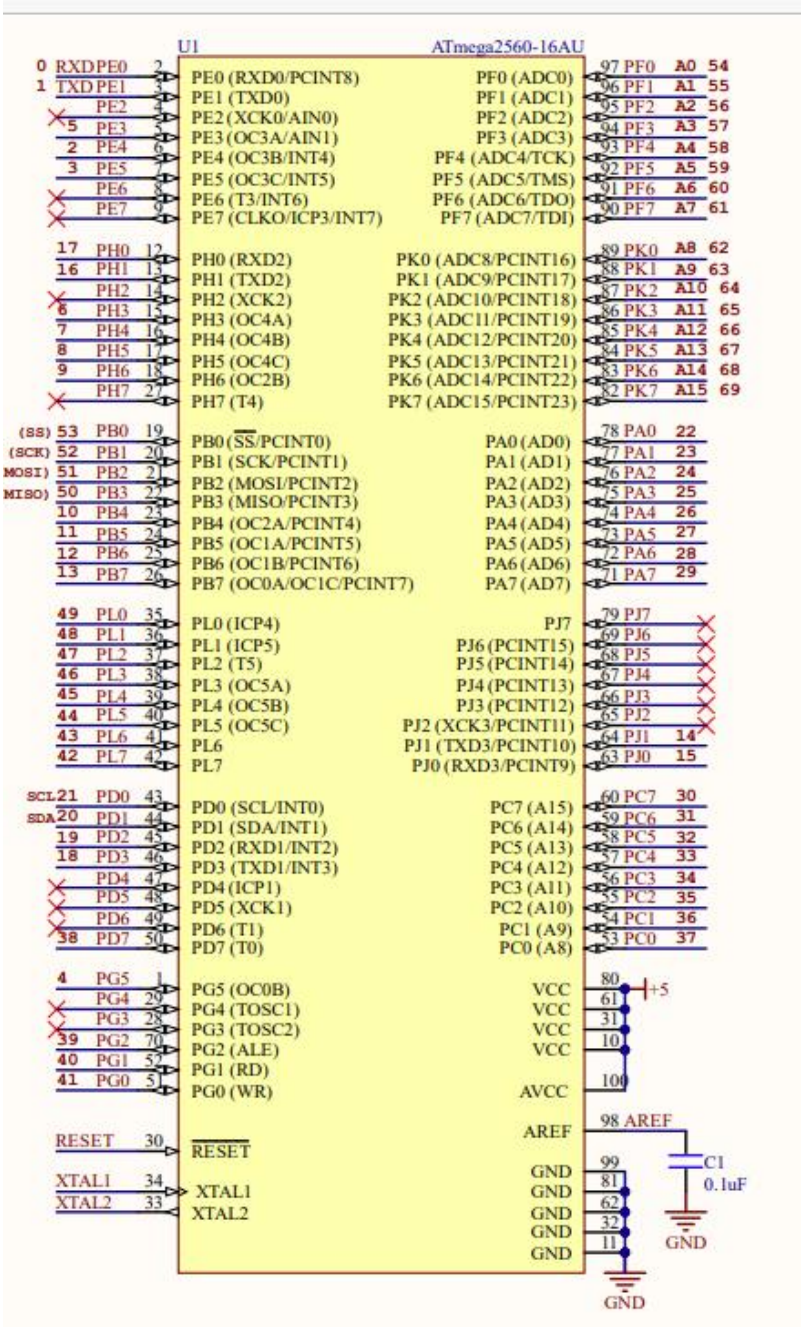


图 5 CPU 引脚图

（2）传感器接口资源分配

智能传感系统 12 个传感器接口资源分配如图 6 所示。每个接口具有多种功能，传感器模块可插入符合接口要求的任一接口，图 6 中所示为出厂默认传感器接口。

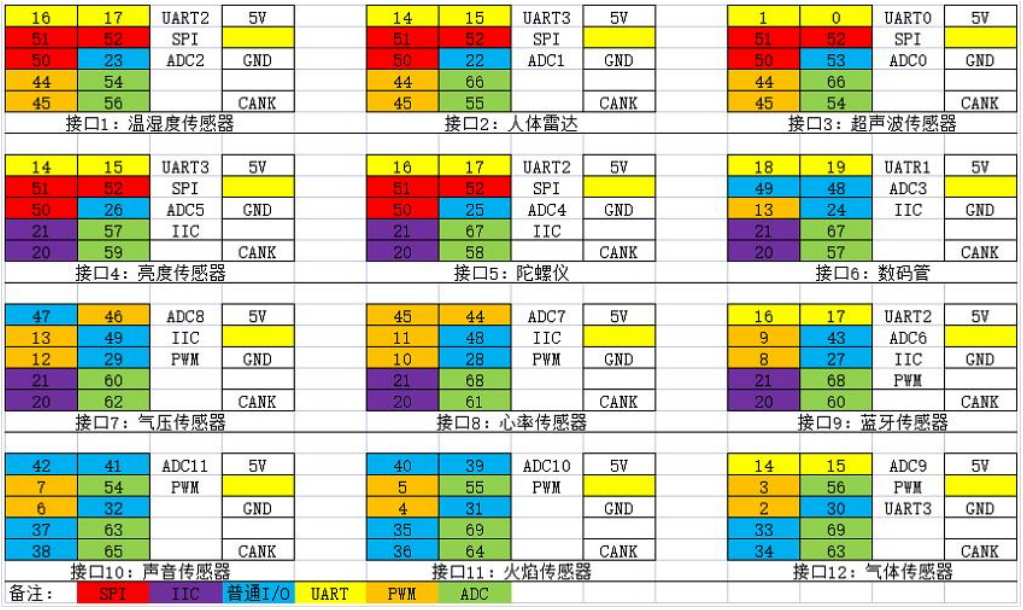
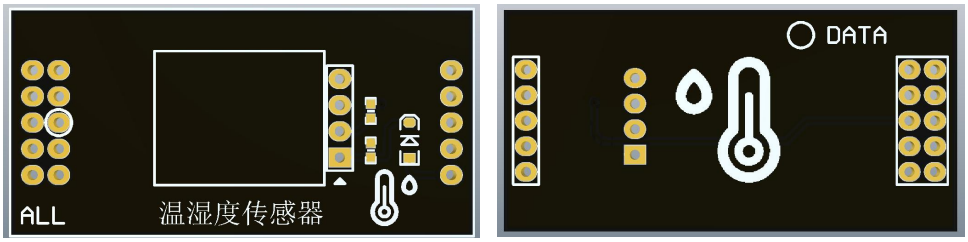


图 6 传感器接口资源分配

例如，温湿度传感器默认插入接口 1 位置，其电路板正反面如图 7 所示。正面图中“ALL”代表温度传感器是通用型传感器，可接插 12 个接口中任一接口。排针中白色“〇”圈出温度传感器所使用的接口引脚。反面图中白色“〇”代表温度传感器所使用的数据类型为 DATA。若温度传感器插入第 1 个接口，则温度传感器的 DATA 引脚连接主板的 23 号引脚，CPU 从 23 号引脚读取温湿度数据；若温度传感器插入第 2 个接口，则温度传感器的 DATA 引脚连接主板的 22 号引脚，CPU 从 22 号引脚读取温湿度数据。插至其它接口以此类推。



（1）正面　　（2）正面

图 7 温湿度传感器电路板

5. 切换控制

智能传感系统通过图 8 所示的拨码开关控制云台的 2 个舵机、风扇、蜂鸣器、3 个按键、电位器是否可用，拨向上，可用；拨向下，不可用。

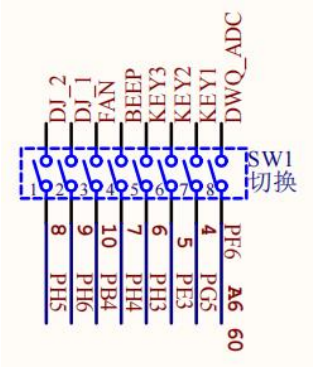


图 8 切换控制拨码开关图

五、实验步骤

插上电源，将拨码开关全部拨向上，按“启动”按钮，运行演示程序，了解智能传感系统的基本功能。演示程序包含温湿度检测、测距仪等子程序，通过按键 key1（↑）、key2（确认）key3（↓）、Reset（退出）选择子程序运行。

六、常见问题

1. 读取传感器数据错误，检查传感器模块是否在出厂默认位置。
2. 不能运行演示程序，检查拨码开关是否全部拨向上，检查芯片中是否烧录演示程序。

Arduino 编程环境的搭建

一、实验目的

（1）了解嵌入式系统基本原理；

（2）熟悉 Linux 系统下 Arduino 编译器的安装；

（3）掌握 Arduino 编译器基本介绍及使用方法；

（4）掌握使用 Arduino 编译器编辑第一个 Arduino 程序。

二、实验内容

嵌入式系统是以应用为中心、以计算机技术为基础、软件硬件可裁剪，且适应系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。嵌入式系统具有体积小、可靠性高、功能强、方便灵活等特点，因而广泛应用于各类电子系统硬件设计。

本实验以 Arduino 为典型嵌入式系统模块，在 Linux 系统中搭建编程环境，并安装 Arduino 编译器及 Arduino 编程中所使用的库，应用 Arduino IDE 编译器上开发示一个例程序，并将程序下载到 Arduino 开发板上进行测试，通过串口监视器显示。

三、实验环境

硬件环境	JETSON XAVIET NX 计算力 21TOPS, 内存 8GB
操作系统	Linux ARM64
编译环境	Arduino IDE
实验设备	人工智能实验箱智能传感系统
实验配件	键盘、鼠标、电源 12V/4A

四、实验原理

1.嵌入式系统

嵌入式系统定义有很多，以下是常用的 3 种定义：

（1）国内普遍认同的嵌入式系统定义：以应用为中心、以计算机技术为基础、软件硬件可裁剪，且适应系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。

（2）IEEE 的定义：用于控制、监视或者辅助操作机器和设备的装置。

（3）其他定义：以提高对象体系智能性、控制力和人机交互能力为目的，通过相互作用和内在指标评价的，嵌入对象体系的专用计算机系统。

嵌入式系统优点：体积小、可靠性高、功能强、方便灵活。

嵌入式系统缺点：系统资源有限、内核小、处理能力有限、实现的功能有限、软件

对硬件的依赖性高、软件的可移植性差、对操作系统的可靠性要求较高、对开发人员的专业性要求较高。

Arduino 基于 Linux 系统搭建编程环境，是在 Linux 系统上安装 Arduino 编译器及 Arduino 编程中所使用的库。熟悉如何在 Arduino IDE 上开发程序，并将程序下载到 Arduino 开发板上进行测试等基本使用方法。

嵌入式系统有微处理器、存储器、输入/输出设备及通信与扩展接口，除了以上四大部分，嵌入式系统还包括时钟与总线、内存管理、看门狗和供电与能耗等要素。

Arduino 是一个简单易用的开源电子平台。Arduino 开发板可读取开关或传感器的数据，并控制电机、LED 灯等。通过对 Arduino 板上控制器进行软件编程，可控制 Arduino 实现所需要的功能。软件开发环境是基于 Processing 的 Arduino IDE。对初学者来说，Arduino 软件简单易学，对有经验的用户来说又足够灵活，Arduino 已经被应用在成千上万的工程和应用系统中。它可运行在 MacOS、Windows 和 Linux 操作系统中。Arduino 简化了微控制器的工作过程，具有以下优点：

价格便宜、跨平台、简单清晰的编译环境、开源和可扩展软件、开源和可扩展硬件。

2.Arduino 软件开发

在 Arduino 官方网站（https：//www.arduino.cc/en/software）下载对应最新版 Arduino IDE，实验平台使用的系统是 Linux ARM 64bit，所以下载 Linux ARM 64bit 的 Arduino IDE，实验开发中使用的是 Arduino IDE 1.8.15，如下图所示：

图 1　Linux ARM 64bit Arduino IDE 1.8.15 下载界面

下载完后会有一个 Arduino IDE 1.8.15 的压缩包文件，将压缩文件解压到当前文件夹，如下图所示：

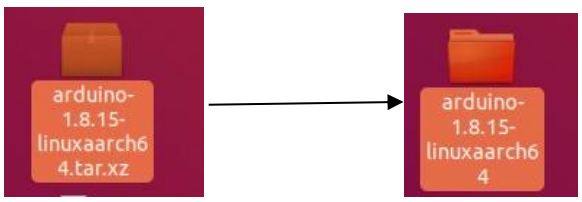


图 2　Arduino IDE 1.8.15 压缩包文件解压

解压完文件夹，进入到文件夹中，鼠标右键打开终端，运行指令 sudo chmod +x install.sh && sudo ./install.sh，等待安装完毕，安装完毕桌面会有 Arduino IDE 图标，鼠标双击即可开始使用 arduino 了。安装如下图所示：

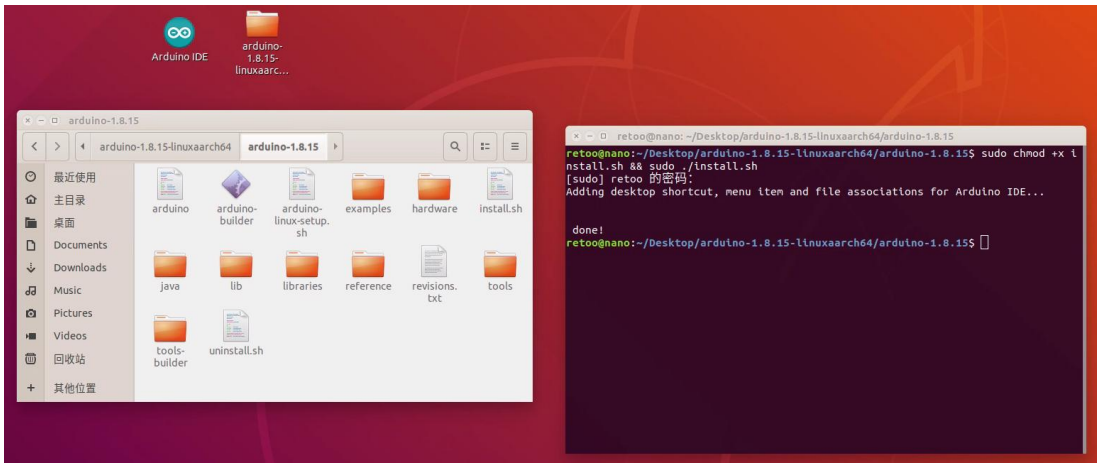


图 3　Arduino IDE 1.8.15 安装

Arduino 开发流程：

（1）连接 Arduino 板

（2）打开项目

（3）选择板的类型和通信端口

（4）下载程序

Arduino 数据类型转换函数：

（1）数据类型转换函数：byte()，char()，float()，int()，

（2）数据类型：String，String 类，array，bool，double，float，int，long，short，unsigned char，unsighed int，unsigned long，void，word

Arduino 程序结构：

（1）setup()

初始化函数，只运行一次。

（2）loop()

循环执行 loop 里的语句，实现对 Audrino 板的控制。

Arduino 控制语句：break，continue，do...while，while，if，if...else，for，return，switch...case

Arduino 其他语句：#define，#include，/* */，//，{}

Arduino 运算符：%，*，+，-，/，=

Arduino 关系运算符!=，<，<=，==，>，>=

Arduino 位运算符&，|，~，^，<<，>>

Arduino 复合运算符+=，-=，*=，/=，&=，|=，^=，++，--

Arduino 指针运算符&，*

Arduino 编程常用函数：

（1）时间函数：

delay()

功能：延长一段时间（单位为 ms）。

语法格式：delay(ms)。

参数说明：ms，延时的毫秒数（usigned long 类型）。

返回值：无。

（2）数学函数：

abs()

功能：取绝对值。

语法格式：abs(x)。

参数说明：x，整数。

返回值：若 x 大于等于 0，返回 x；若 x 小于 0，返回-x。

constrain()

功能：将值归一化在某个范围内。

语法格式：constrain(x,a,b)。

参数说明：x，需要归一化的数据；a，数据下限；b，数据上限。三者均为任意数据类型。

返回值：若 x 在 a 和 b 之间，返回 x；若 x 小于 a，返回 a；若 x 大于 b，返回 b。

max()

功能：计算两个数中的较大者。

语法格式：max(x,y)。

参数说明：x，第一个数据；y，第二个数据。二者可以为任意数据类型。

返回值：两个数中的较大者。

min()

功能：计算两个数中的较大者。

语法格式：min(x,y)。

参数说明：x，第一个数据；y，第二个数据。二者可以为任意数据类型。

返回值：两个数中的较小者。

pow()

功能：计算一个数的幂。幂可以是一个分数幂，方便用于求一个数或曲线的指数映射。

语法格式：pow(base,exponent)；

参数说明：base，数据（float 类型）；exponent，幂（float 类型）。

返回值：求幂的结果（double 类型）。

sqrt()

功能：计算一个数的平方根。

语法格式：sq(x)。

参数说明：x，实数（任意数据类型）。

返回值：求平方根的结果（double 类型）。

五、实验步骤

1.运行 Arduino IDE

双击桌面图标 Arduino IDE，或者进入 Arduino IDE 安装文件目录，右键选择运行终端，输入./arduino 回车，即可打开 Arduino IDE，如下图所示：

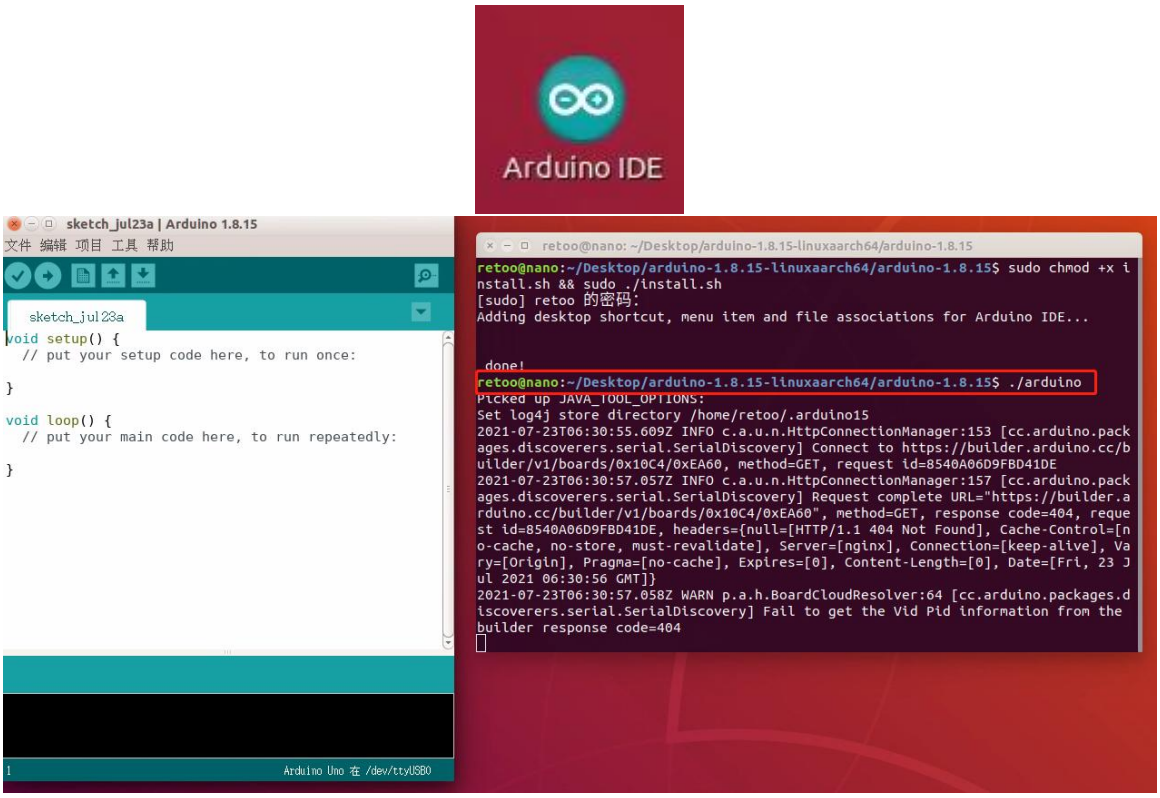


图 4　Arduino IDE 1.8.15 运行

2. Arduino IDE 界面介绍

Arduino IDE 界面分为以下几个功能区“功能菜单栏”，“常用快捷键”，“代码编辑区”，“代码编译输出信息”，“串口监视器快捷键”，如下图所示：

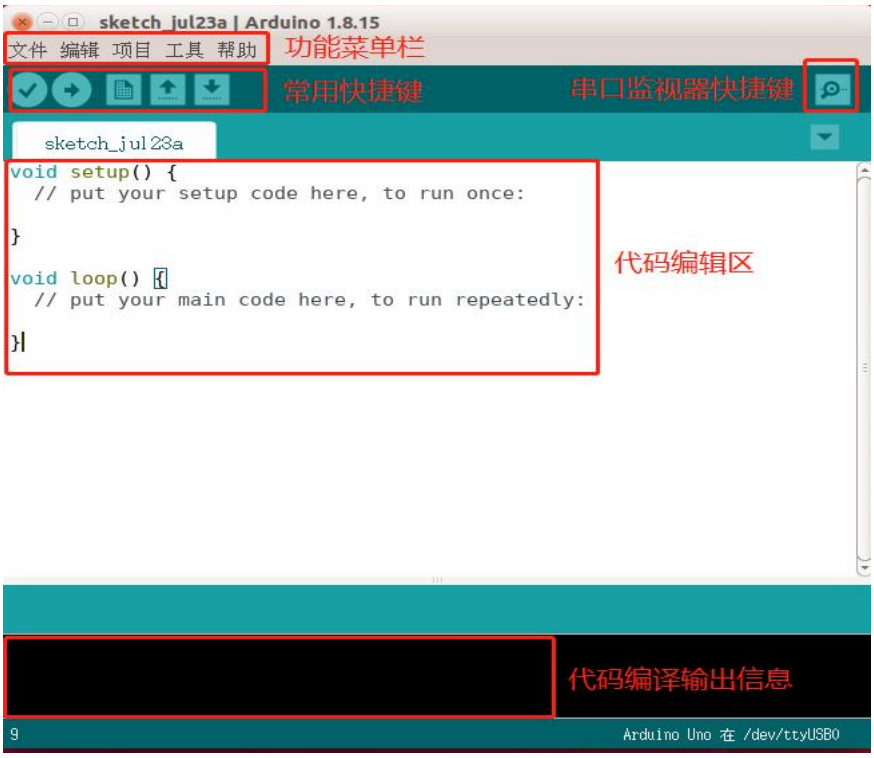


图 5　Arduino IDE 1.8.15 界面介绍

3. Arduino IDE 功能区介绍

（1）文件

文件功能是对我们编程文件的常用操作，有“新建”，“打开”“保存”，“另存为”，“示例”，“页面设置”等常用的操作，如下图所示：

图 6　文件功能介绍

若需要新建一个工程文件，点击新建即可，点击另存为，将文件另存到可选择的路径保存，另存或者保存操作时，需要设置文件的名称，根据自己的工程设置合适的名字。如下如所示，选择路径及输入名称就会在相应的路径下生成一个文件夹，文件夹里面就是我们新建的工程文件***.ino。如下图所示：

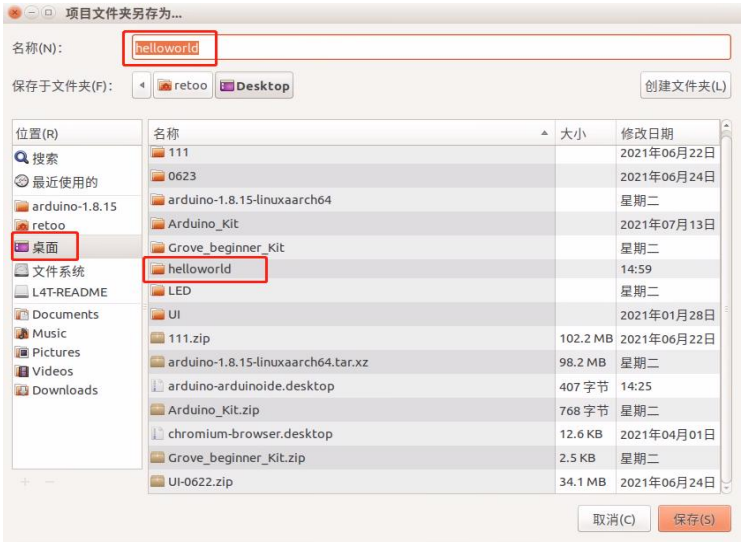


图 7　文件另存操作

在文件的示例中，Arduino 官方提供了很多常用的基本例程，在开发时，可以参考例程进行开发。

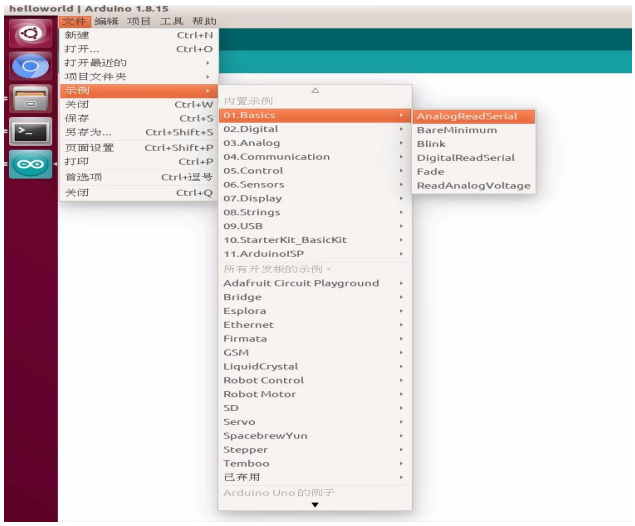


图 8　示例程序

（2）编辑

编辑功能是在编写程序时，对代码的操作，包括注释，粘贴，复制，字号调整等常用的编辑操作。

（3）项目

项目功能包含编译，上传，加载库，添加文件等开发工具，编译是将编写的代码在 Arduino IDE 中进行编译生成 Arduino UNO 开发板可执行的二进制文件。上传是将编译没有问题的二进制文件下载到 Arduino UNO 开发板的处理器中。添加库是添加开发中常用的第三方库文件。添加文件可以通过自己编写库文件，也可以下载库文件进行加载安装。

（4）工具

工具功能包含格式调整，项目打包，库管理，串口监视器，开发板选择，串口选择

等。我们在开发时，要选择对应的开发板的，和所连接的串口号，实验箱中所使用的开发板是 ArduinoUNO 开发板。

4. HelloWorld 程序编写

在编写第一个 Arduino 程序之前，先介绍一下 arduino 程序开发的基本知识，arduino 程序是基于 c 语言和 c++进行开发，具有面向对象的编程思想。Arduino 开发程序基本格式分为 setup{}函数和 loop{}函数，setup{}是进行一些初始化相关的操作，比如初始化串口，初始化 Arduino UNO 开发板上的 I/O 口等操作，loop{}是一个循环体函数，开发板上电就会循环往复的执行 loop{}函数中的内容，以下是 Arduino 的第一个开发程序作为示例。

（1）打开桌面上的 Arduino IDE 软件，单击新建“  ”，再单击保存“  ”，输入名称：×××，即同时创建一个项目文件夹：×××和程序：×××.ino，项目文件夹名称和程序名相同，可包含数字、英文字母，不能包含汉字、空格。

参考代码：

```
String strword = "Hello World";//定义输出的字符串
void setup() {
    Serial.begin(9600);//初始化串口，设置波特率9600
}
void loop() {
    Serial.println(strword);//串口监视器显示字符
    delay(1000);//延时1秒
}
```

（2）点击快捷键编译按钮“  ”，ArduinoIDE 会将编写的程序进行编译，并在输出窗口进行显示，操作如下图所示：

图 9 编译代码

如果程序编译没有问题就会在显示编译完成，并显示当前程序所占用的内存空间大小。

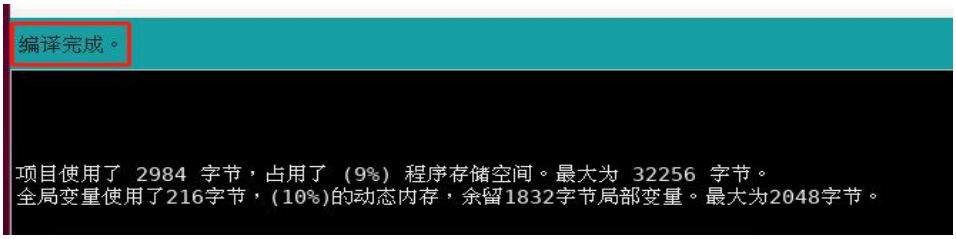


图 10　编译结果输出

（3）点击快捷键上传按钮“  ”上传程序，上传完毕即开始运行程序。

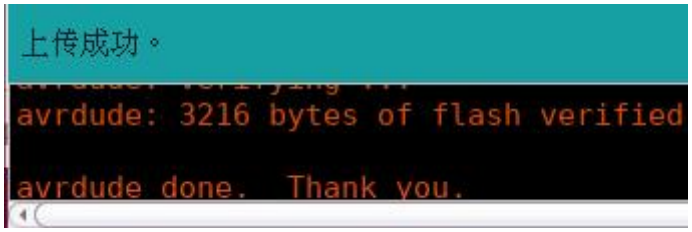


图 11 上传结果输出

（4）单击右上角“  ”打开串口监视器，可以看到在串口监视器中输出的结果信息，如图 12 所示：

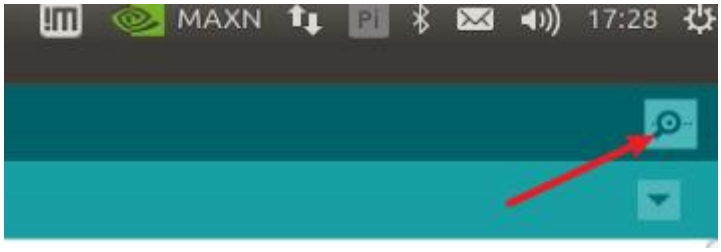


图 12 打开串口监视器

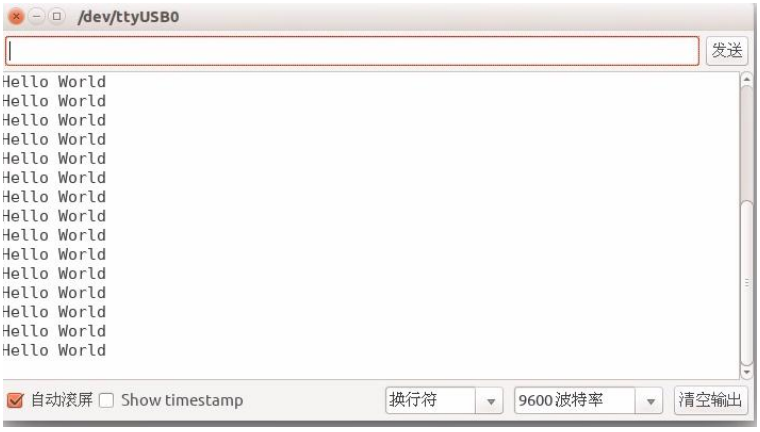


图 13　串口监视器输出打印结果

实验三 OLED 显示

一、实验目的

（1）掌握智能传感系统 OLED 接口电路原理；

（2）掌握 OLED 显示屏的工作原理；

（3）掌握 OLED 显示屏显示方法；

二、实验内容

本实验要求在熟悉 OLED 接口电路、工作原理的基础上，编程实现在 OLED 显示屏显示“hello word”。

三、实验环境

硬件环境	JETSON XAVIET NX 计算力 21TOPS, 内存 8GB
操作系统	Linux ARM64
编译环境	Arduino IDE
实验设备	人工智能实验箱智能传感系统
实验配件	键盘、鼠标、电源 12V/4A

四、实验原理

智能传感系统主控板原理见本课程实验一 智能传感系统认知。

1. OLED 显示屏

采用 SH1106 OLED 显示屏，其参数见表 1。通过 IIC 进行通信，其接口电路如图 1 所示。

表 1 OLED 屏参数

尺　寸	1.3 英寸
材　料	PM OLED
分辨 率	128*64
控制芯片	SH1106(如有需要 SSD1306 的客户请与客服联系) SH1106 的起始地址为 0x02 列；SSD1306 为 0x00
显示区域	29.42 × 14.7 (mm)
PCB 尺寸	35.4x33.5 (mm)
接口类型	IIC 接口（两个设备地址可以切换）
显示颜色	白色，蓝色，两种颜色可选，下单时请指定颜色
工作电压	3.3～5V
像素尺寸	0.21 × 0.21 (mm)
像素间距	0.23 × 0.23 (mm)
工作电流	正常工作时电流在 20ma 左右休眠时在 ua 级电流
管脚数量	4
视角方向	全视角
工作温度	-40～70 度

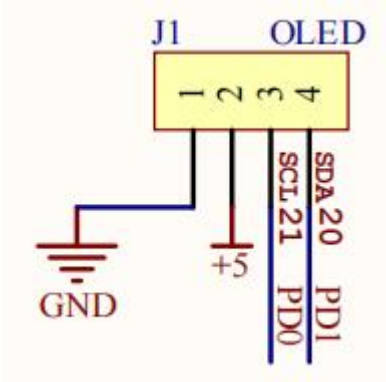


图 1 OLED 屏 IIC 接口电路

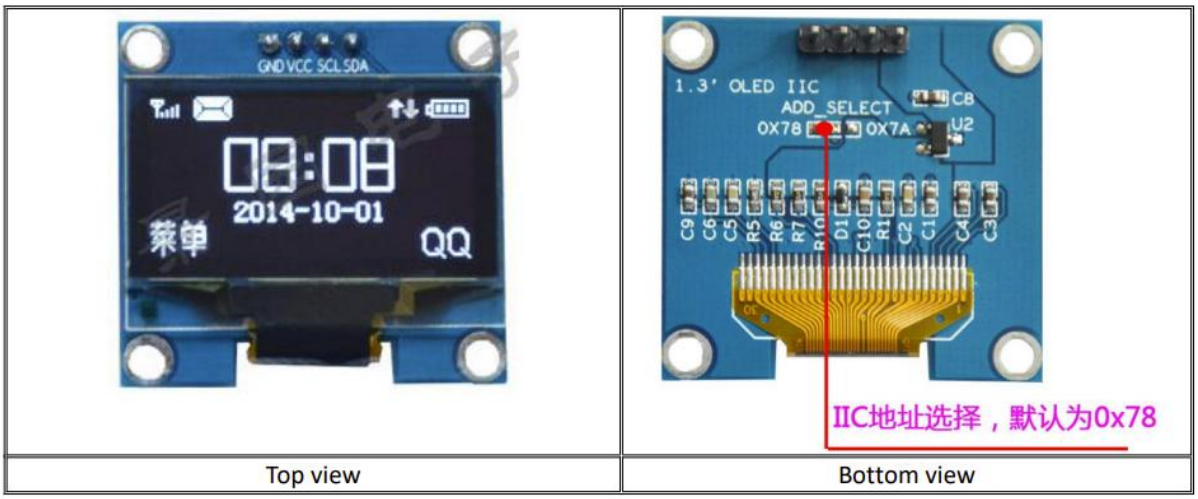


图 2 OLE 显示屏外观

2.相关函数

（1） U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE)—— 选择 1.3 寸 SH1106 OLED 显示屏；

（2） u8g2.begin()——构造 u8g2 模式：初始化显示器，重置清屏，唤醒屏幕；

（3） u8g2.clearBuffer()——清缓冲区；

（4） u8g2.sendBuffer()——将缓冲区数据发送给示显器，发刷新消息；

（5） u8g2.setFont()——设置字体；

（6） u8g2.setCursor()——设置光标处；

（7） u8g2.print()——打印文本字符；

（8） u8g2.drawStr()——绘制字符串；

（9） Serial.begin()——设置串口波特率，启动串口

（10） Serial.println()——串口监视窗口显示数据

（11）pinMode()——设置 I/O 引脚输入/输出模式

（12） digitalRead()——读取 I/O 引脚信号

（13） digitalWrite()——置 I/O 引脚为高/低电平

五、实验步骤

1. 打开桌面上的 Arduino IDE 软件，单击新建“”，再单击保存“”，输入名称：×××，即同时创建一个项目文件夹：×××和程序：×××.ino，项目文件夹名称和程序名相同，可包含数字、英文字母，不能包含汉字、空格。

2. 添加库函数和头文件

添加库函数和头文件，选择显示屏型号。

参考代码:

```
#include <Arduino.h>
#include <U8g2lib.h>
#ifdef U8X8_HAVE_HW_SPI
#include <SPI.h>
#endif
#ifdef U8X8_HAVE_HW_I2C
#include <Wire.h>
#endif
U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);//选择 1.3 寸 SH1106 OLED 显示屏
```

3. 编写初始化程序

初始化显示屏。

参考代码:

```
void setup(void) {
  u8g2.begin();//构造 u8g2 模式：初始化显示器，重置清屏，唤醒屏幕
}
```

4. 编写主程序

清显示屏缓冲区，选择合适的字体，显示字符串“Hello World”。

参考代码:

```
void loop(void) {
  u8g2.clearBuffer();                  // 清缓冲区
  u8g2.setFont(u8g2_font_ncenB08_tr); // 设置字体
  u8g2.drawStr(0,10,"Hello World!");   // 绘制字符串“Hello World”
  u8g2.sendBuffer();                   // 将缓冲区数据发送给示显器，发刷新消息
  delay(1000);
}
```

5. 单击“编译(✓)”以检查语法错误，单击 “上传(→)”代码

6.运行程序，显示运行结果。

图 3 实际运行效果改图

实验一 语音处理系统认知

一、实验目的

（1）了解语音处理系统的硬件布局。

（2）熟悉语音处理系统的硬件系统结构；

（3）了解语音处理系统的硬件系统原理；

（4）熟悉语音处理系统的端口资源分配；

二、实验内容

本实验要求在熟悉语音系统硬件结构的基础上，掌握系统的硬件系统原理和端口资源及系统的功能。

三、实验环境

硬件环境	JETSON XAVIET NX 计算力 21TOPS, 内存 8GB
操作系统	Linux ARM64
编译环境	MicroPython -USB
实验设备	人工智能实验箱智能语音处理系统
实验配件	键盘、鼠标、电源 12V/4A

四、实验原理

1. 语音处理系统认知

语音处理开发板如图 1 所示，包含以下主要部件：

（1） 采用 64 位双核 K210 处理器；

（2） 7 个 MEMS 麦克风；

（3） 12 个 LED；

（4） 1 个 USB ；

（5） 1 个 SD 卡槽；

（6） 1 个触摸电源按键；

（7） 128Mbit Flash；

（8） 2 个扬声器

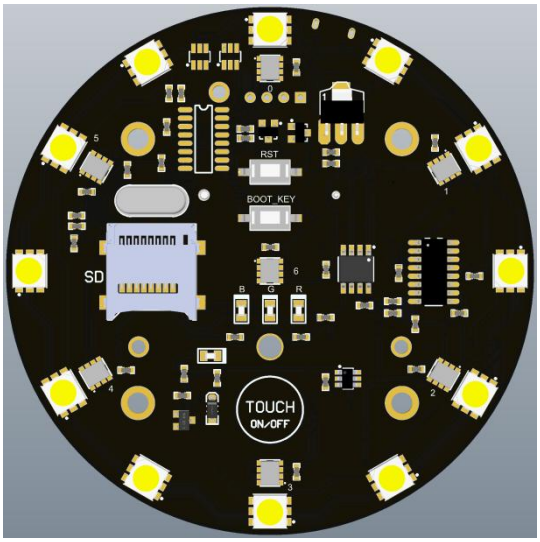


图 3.1.1　K210 语音开发板

语音处理系统采用 K210 作为 CPU，功能非常很强大，芯片内置 64 位双核处理器，拥有 8M 的片上 SRAM，K210 基本参数见表 1。

表 1　K210 芯片基本参数

内核	RISC-V Dual Core 64bit, with FPU
主频	400MHz （可超频至 600MHz）
SRAM	内置 8M Byte
图像识别	QVGA@60fps/VGA@30fps
语音识别	麦克风阵列(8mics)

语音模块由 K210 作为核心单元组成，功能非常很强大，芯片内置 64 位双核处理器，拥有 8M 的片上 SRAM。

2. 语音处理系统硬件原理框图

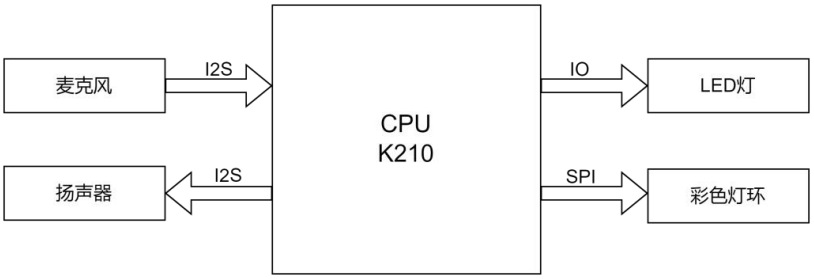


图 3　智能传感系统硬件原理框图

3. 语音处理系统硬件原理图

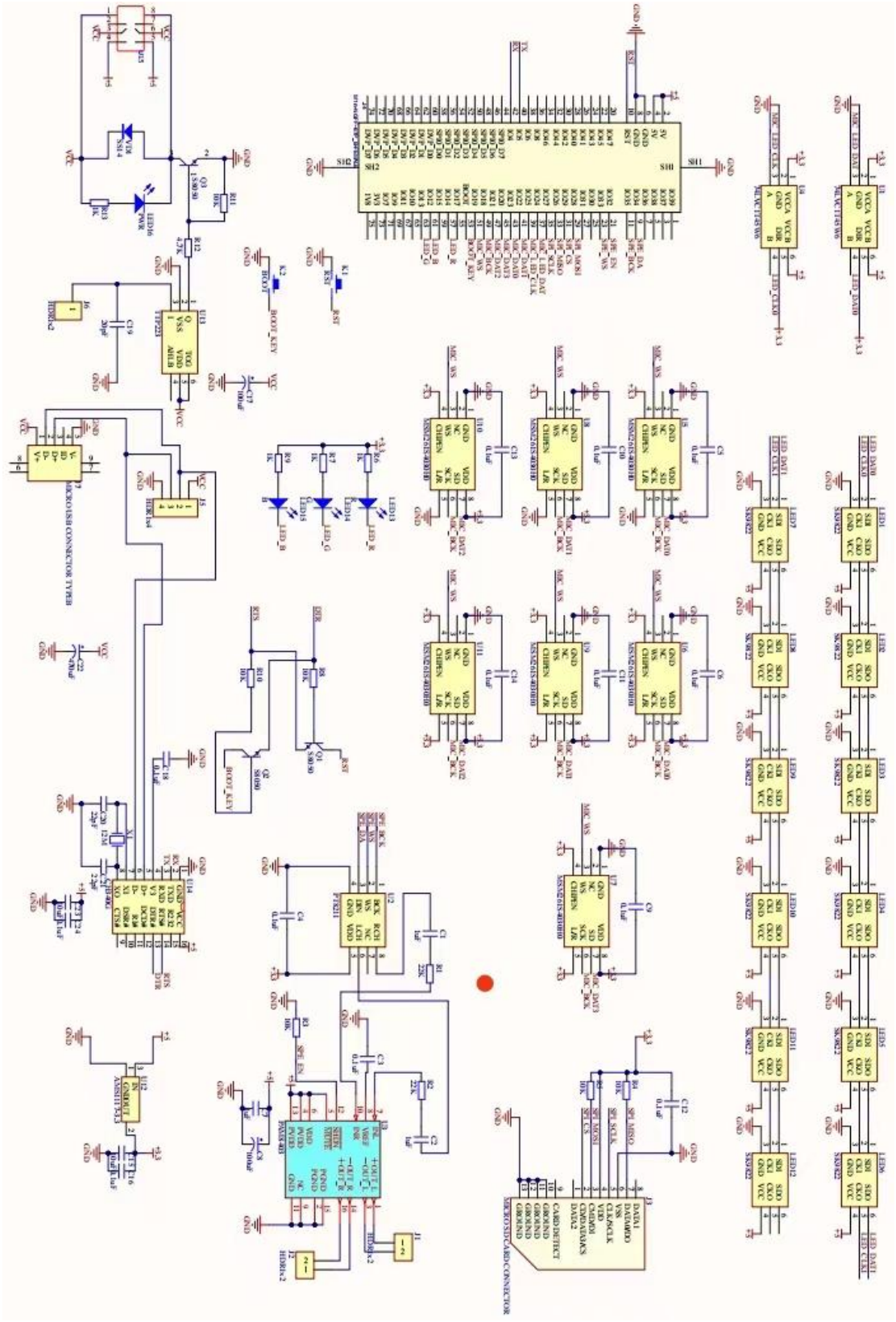


图 4 语音处理系统硬件原理图

4. 语音处理系统端口资源分配

（1）CPU K210 引脚图

CPU K210 引脚图如图 5 所示。

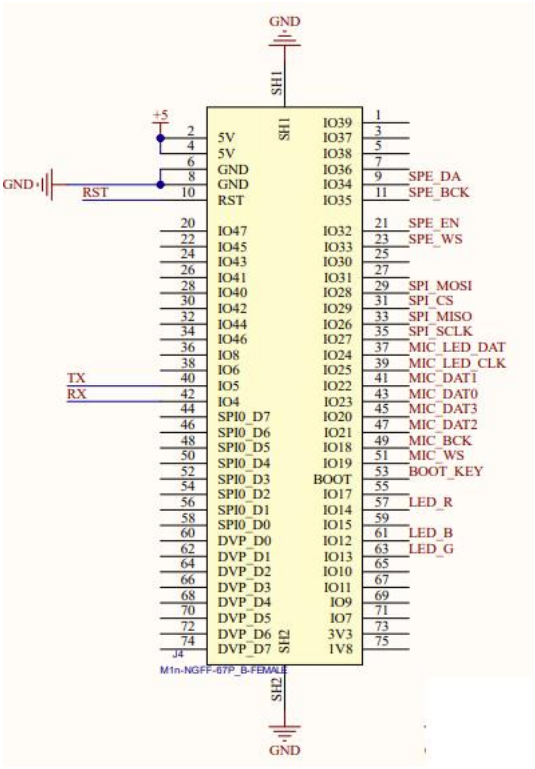


图 5 CPU 引脚图

（2）端口资源分配

语音处理系统端口资源分配见表 1。

表 1 端口资源分配

CPU 引脚	接口引脚标号	接口引脚名称
IO12	LED_B	蓝色 LED 灯
IO13	LED_G	绿色 LED 灯
IO14	LED_R	红色 LED 灯
IO18	MIC_BCK	麦克风 I2S 接口串行位时钟信号
IO19	MIC_WS	麦克风 I2S 接口帧时钟信号
IO20	MIC_DAT3	麦克风 U7 I2S 接口数据线
IO21	MIC_DAT2	麦克风 U10 和 U11 I2S 接口数据线
IO22	MIC_DAT1	麦克风 U8 和 U9 I2S 接口数据线
IO23	MIC_DAT0	麦克风 U5 和 U6 I2S 接口数据线
IO24	MIC_LED_DAT	灯环 SPI 接口 MOSI 数据线
IO25	MIC_LED_CLK	灯环 SPI 接口同步时钟

IO32	SPE_EN	扬声器功放使能信号
IO33	SPE_WS	扬声器 I2S 接口帧时钟信号
IO34	SPE_DA	扬声器 I2S 接口数据线
IO35	SPE_BCK	扬声器 I2S 接口串行位时钟线

五、实验步骤

插上电源，触摸语音处理系统开发板上的“touch”按钮，给语音处理系统上电，灯环亮起。

六、常见问题

1. 连接不上，检查电源是否打开，USB 端口选择是否正确。
2. 运行程序没反应，按“RST”键，手动复位。

实验二 LED 灯控制

一、实验目的

(1)了解 GPIO(通用输入/输出)的特点；

(2)掌握 LED 灯接口电路的原理；

(3)掌握 GPIO 的使用方法。

二、实验内容

本实验要求在掌握 GPIO 特点和 LED 灯接口电路原理的基础上，编程实现红、绿、蓝三色 LED 灯循环点亮，首先配置 GPIO 引脚，然后初始化，最后循环点亮三色 LED 灯。

三、实验环境

硬件环境	JETSON XAVIET NX 计算力 21TOPS, 内存 8GB
操作系统	Linux ARM64
编译环境	Jupyter lab MicroPython
实验设备	人工智能实验箱智能语音处理系统
实验配件	键盘、鼠标、电源 12V/4A

四、实验原理

1.LED 灯接口电路

LED 灯接口电路如 1 所示，其蓝色 LED_B、绿色 LED_G、红色 LED_R、分别连接 CPU 的第 IO12、IO13、IO14 通用输入/输出（GPIO）引脚上。

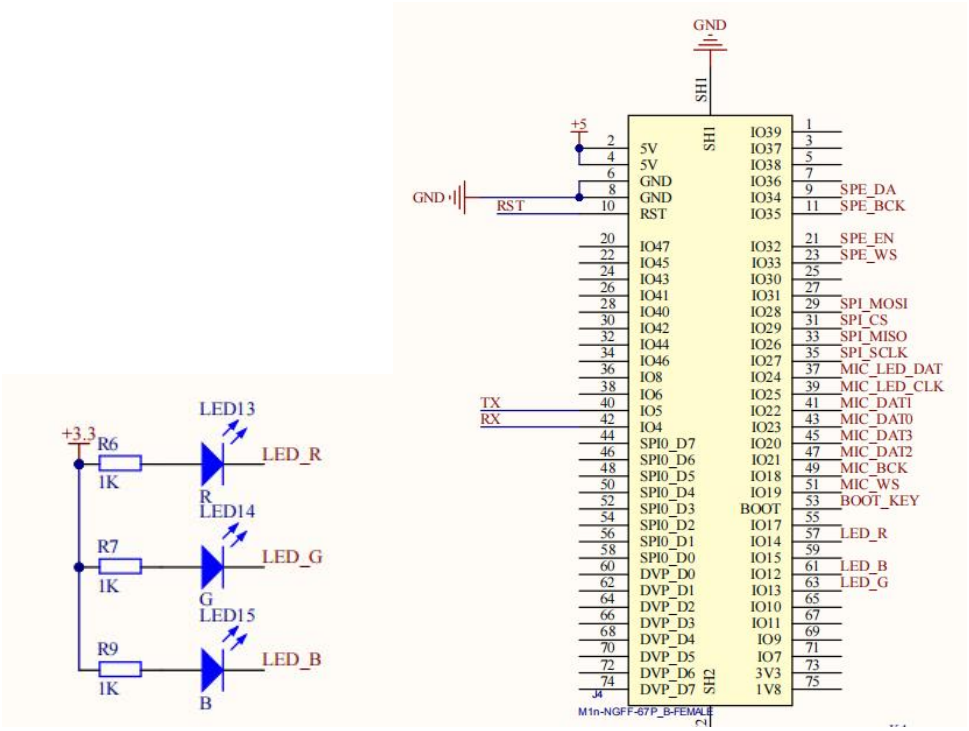


图 1 LED 灯接口电路

其蓝色 LED_B、绿色 LED_G、红色 LED_R、分别连接 CPU 的第 IO12、IO13、IO14 通用输入/输出（GPIO）引脚上。

LED 灯使用引脚：

IO12：LED_B

IO13：LED_G

IO14：LED_R

2. GPIO 简介

General Purpose Input Output （通用输入/输出）简称为 GPIO，分为高速 GPIO(GPIOHS) 和通用 GPIO。CPU K210 上有 32GPIOHS 和 8 个通用 GPIO。

高速 GPIO 具有如下特点：

（1）可配置输入输出信号；

（2）每个 IO 具有独立中断源；

（3）中断支持边沿触发和电平触发；

（4）每个 IO 可以分配到 FPIOA 上 48 个管脚之一；

（5）可配置上下拉，或者高阻态。

通用 GPIO 具有如下特点:

（1）8 个 IO 使用一个中断源；

（2）可配置输入输出信号；

（3）可配置触发 IO 总中断，边沿触发和电平触发；

（4）每个 IO 可以分配到 FPIOA 上 48 个管脚之一。

注意:表 1 中 GPIOHS 默认已经被使用， 程序中如非必要尽量不要使用：

表 1 K210 默认使用的 GPIOHS

GPIOHS	功能
GPIOHS31	LCD_DC
GPIOHS30	LCD_RST
GPIOHS29	SD_CS
GPIOHS28	MIC_LED_CLK
GPIOHS27	MIC_LED_DATA

3. 相关函数

（1）class GPIO(ID,MODE,PULL,VALUE)

功能：通过指定的参数新建一个 SPI 对象。

参数 ID：使用的 GPIO 引脚(一定要使用 GPIO 里带的常量来指定)。

参数 MODE：GPIO 模式（GPIO.IN—输入模式，GPIO.OUT—输出模式）。

参数 PULL：GPIO 上下拉模式（GPIO.PULL_UP—上拉，GPIO.PULL_DOWN—下拉，GPIO.PULL_NONE—既不上拉也不下拉（高阻））

（2）GPIO.value([VALUE])

功能：修改/读取 GPIO 引脚状态。

参数[value]：可选参数，如果此参数不为空，则返回当前 GPIO 引脚状态。

返回值：如果 [value] 参数不为空，则返回当前 GPIO 引脚状态。

四、实验步骤

1.打开语音开发板电源

触摸 K210 语音开发板上的 touch 按钮，启动语音开发板电源。

2.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在“实验”文件夹空白处单击鼠标右键，选择“打开终端”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 MicroPython -USB，进入程序编辑器。

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为所做实验名。

3.连接 USB 端口

引入连接 MicroPython -USB 语音开发板的接口。

参考代码:

```
#连接的usb端口
%serialconnect to --port=/dev/user_voice --baud=115200
```

4.导入函数库，编写代码

（1）导入 GPIO 函数库。

参考代码:

```
from Maix import GPIO
from fpioa_manager import fm
import utime
```

（2）定义 LED 灯的 IO 连接，并初始化。

参考代码:

```
# 将 LED 外部 IO 注册到内部 GPIO，K210 引脚支持任意配置
fm.register(12, fm.fpioa.GPIO0)   # LED_B
fm.register(13, fm.fpioa.GPIO1)   # LED_G
fm.register(14, fm.fpioa.GPIO2)   # LED_R

# 构建 LED 对象，并初始化输出高电平，关闭 LED
LED_B = GPIO(GPIO.GPIO0, GPIO.OUT, value=1)
LED_G = GPIO(GPIO.GPIO1, GPIO.OUT, value=1)
LED_R = GPIO(GPIO.GPIO2, GPIO.OUT, value=1)

# 定义 LED 数组，方便循环语句调用
LED = [LED_B, LED_G, LED_R]
```

（3）循环点亮 LED 灯

参考代码:

```
while True:
    for i in range(0, 3):
        LED[i].value(0)   # 点亮 LED
        utime.sleep(1)
        LED[i].value(1)   # 关闭 LED
```

五、操作流程

单击“ ▶▶ ”，运行此 jupyter lab 程序，可看到语音板上的蓝、绿、红灯循环点亮熄灭。

实验三 麦克风阵列灯环控制

一、实验目的

(1)熟悉 SPI 的通信原理；

(2)掌握灯环控制接口电路的原理；

(3)掌握 SPI 的使用方法。

二、实验内容

本实验要求在掌握 SPI 通信原理和灯环控制接口电路的基础上，编程配置灯环 SPI 引脚并初始化，编写驱动程序代码，最后循环点亮 LED 灯。

三、实验环境

硬件环境	JETSON XAVIET NX 计算力 21TOPS, 内存 8GB
操作系统	Linux ARM64
编译环境	Jupyter lab MicroPython
实验设备	人工智能实验箱智能语音处理系统
实验配件	键盘、鼠标、电源 12V/4A

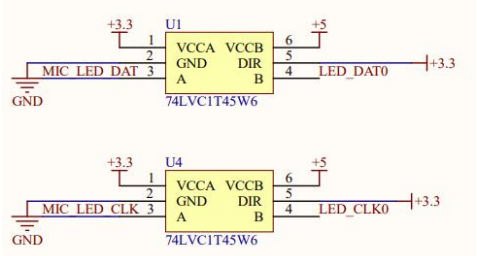
四、 实验原理

1. 灯环接口电路

灯环由 12 个 SK9822 组成，SK9822 是一款双向传输三通道（RGB）驱动控制电路与发光电路于一体的智能外控 LED 光源，采用双输出方式，DATA 数据及同步的 CLK 信号，使串行级联的各芯片输出动作同步。

灯环接口电路如图 1 所示，采用 SPI 接口。两个 74LVC1T45W6 电平转换芯片将 CPU 3.3V TTL 电平 IO24(MIC_LED_DAT)、IO25（MIC_LED_CLK）转换成 5V TTL 电平 LED_DAT0、LED_CLK0，再分别连接至 LED1（SK9822）的 SDI 和 CKI 引脚，各 LED 之间串行级联。其中 SDI——SPI 数据引脚，CKI——SPI 同步时钟引脚。

CPU 通过 SPI 向 LED1 发送数据，12 个 LED 串行移位输出显示。



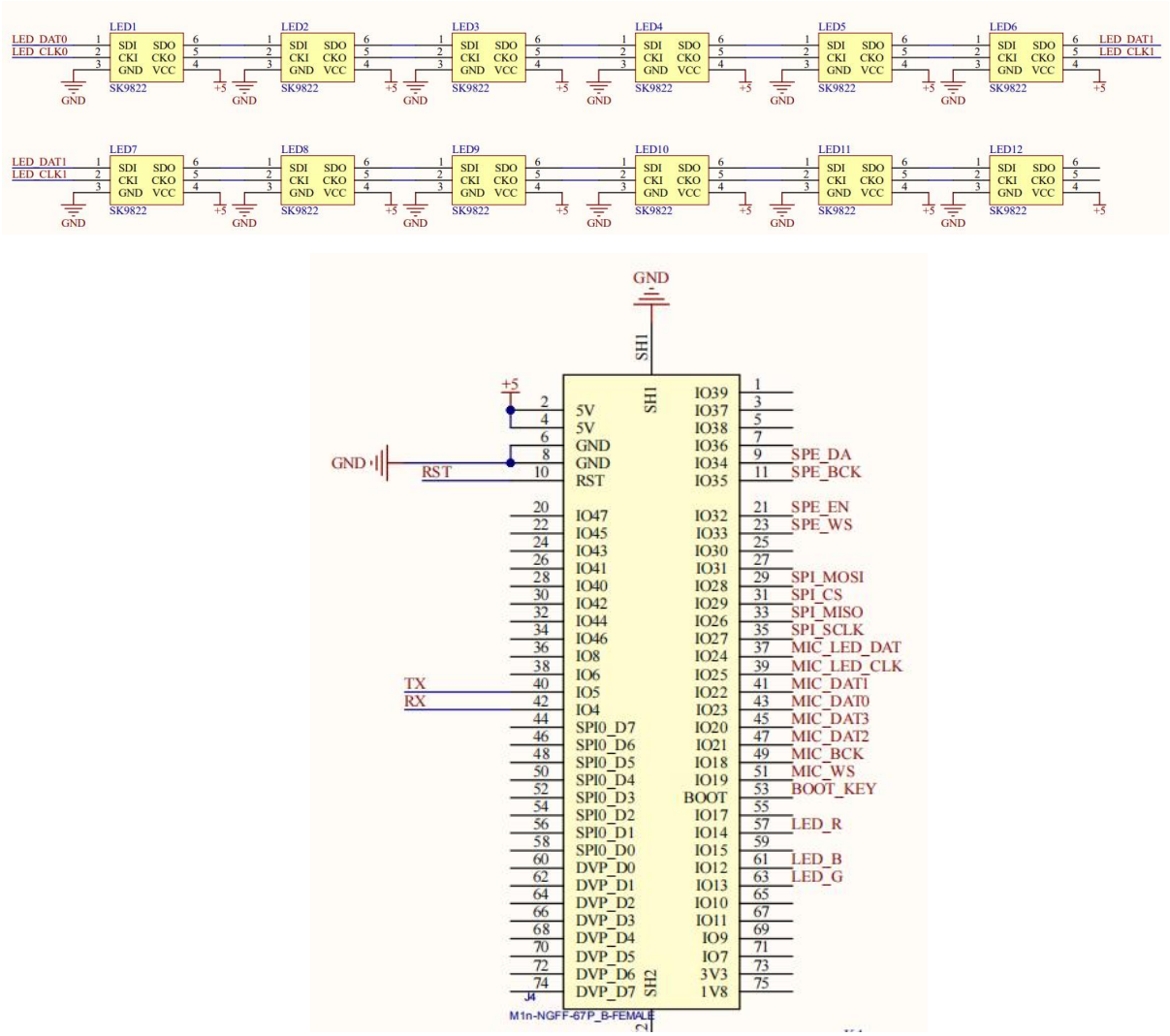


图 1 灯环控制接口电路

灯环使用引脚：

IO24：MIC_LED_DAT

IO25：MIC_LED_CLK

2.SPI 接口

SPI（Serial Peripheral Interface）是一个同步串行协议，由主机和从机组成。SPI 采用标准 4 线模式：SCK（SCLK）、CS（片选）、MOSI（主出从入）、MISO 4（主入从出）。

K210 上的 SPI 具有以下特征：

（1）共有 4 个 SPI 设备，其中 SPI0 、SPI1、 SPI3 只能工作在主机模式下， SPI2 只能工作在从机模式， 在 MaixPy 上，SPI3 已经用来连接了 SPI Flash 作为保留硬件资源。

（2）支持 1/2/4/8 线全双工模式， 在 MaixPy 中，目前只支持标准（摩托罗拉）4 线全双工模式（即 SCK，MOSI，MISO，CS 四个引脚）；

（3）最高传输速率 45M：1/2 主频，约 200Mbps；

（4）支持 DMA；

（5）4 个可配置任意引脚的硬件片选。

3. 相关函数

（1）spi = machine.SPI(id, mode=SPI.MODE_MASTER, baudrate=10000000, polarity=0, phase=0, bits=8, firstbit=SPI.MSB, sck=25,mosi=24,miso,cs0,cs1,cs2,cs3)

功能：通过指定的参数新建一个 SPI 对象;

参数 id：SPI ID，取值范围[0,4]，目前只支持 0 和 1 、4 ，并且只能是主机模式，2 只能作为从机，目前未实现，3 保留, 4 使用软模拟 SPI（.SPI_SOFT）。

参数 mode：SPI 模式，MODE_MASTER 或者 MODE_MASTER_2 或者 MODE_MASTER_4 或者 MODE_MASTER_8 或者 MODE_SLAVE，目前只支持 MODE_MASTER;

参数 baudrate：SPI 波特率（频率）；

参数 polarity：极性，取值为 0 或 1，表示 SPI 在空闲时的极性，0 代表低电平，1 代表高电平；

参数 phase：相，取值为 0 或 1，表示在时钟的第一个还是第二个跳变沿采集数据，0 表示第一个，1 表示第二个；

参数 bits：数据宽度，默认值为 8，取值范围[4,32]；

参数 firstbit：指定传输采用 MSB 还是 LSB 顺序传输，默认 SPI.MSB；

参数 sck: SCK（时钟）引脚，可直接传引脚数值，取值范围：[0,47]。可以不设置，而是使用 fm 统一管理引脚映射。

参数 mosi: MOSI（主机输出）引脚，可直接传引脚数值，取值范围：[0,47]。可以不设置，而是使用 fm 统一管理引脚映射；

参数 miso: MISO（主机输入）引脚，可直接传引脚数值，取值范围：[0,47]。可以不设置，而是使用 fm 统一管理引脚映射；

参数 cs0: CS0（片选）引脚，可直接传引脚数值，取值范围：[0,47]。可以不设置，而是使用 fm 统一管理引脚映射；

参数 cs1: CS1（片选）引脚，可直接传引脚数值，取值范围：[0,47]。可以不设置，而是使用 fm 统一管理引脚映射；

参数 cs2: CS2（片选）引脚，可直接传引脚数值，取值范围：[0,47]。可以不设置，而是使用 fm 统一管理引脚映射；

参数 cs3: CS3（片选）引脚，可直接传引脚数值，取值范围：[0,47]。可以不设置，而是使用 fm 统一管理引脚映射；

参数 d0~d7：数据引脚，在非标准 4 线模式中使用，目前保留。 可以不设置，而是使用 fm 统一管理引脚映射。

（2）spi.write(buf,cs=SPI.CS0)

功能：发送数据；

参数 buf：bytearray 类型，定义了数据及长度；

参数 cs：选择片选引脚，在初始化时已经为 cs0~cs3 设置了引脚，这里只需要选择 SPI.CS0~SPI.CS3 即可，默认为 SPI.CS0。

四、实验步骤

1.打开语音板电源

触摸 K210 语音开发板上的 touch 按钮启动开发板电源。

2.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在“实验”文件夹空白处单击鼠标右键，选择“打开终端”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 MicroPython -USB，进入程序编辑器。

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为所做实验名。

3.连接的 usb 端口

引入连接 MicroPython -USB 语音开发板的接口。

参考代码:

```
#连接的usb端口
%serialconnect to --port=/dev/user_voice --baud=115200
```

4.导入函数库，编写代码

（1）导入 SPI 函数库。

参考代码:

```
from machine import SPI
import time
```

（2）创建 SPI 对象，定义灯环的 IO 连接，编写驱动程序。

参考代码:

```
spi = SPI(SPI.SPI0, mode=SPI.MODE_MASTER, baudrate=10000000, polarity=0, phase=0,
bits=8, firstbit=SPI.MSB, sck=25,
                mosi=24)   # 创建一个 SPI 对象，设置参数：主机模式，波特率，SPI(彩色灯环)
引脚
def on_led(lun, data):   # 彩色灯环控制(亮度，颜色)
    spi.write(0x00)
    spi.write(0x00)
    spi.write(0x00)
```

```
        spi.write(0x00)
        for i in range(12):
            spi.write(0xe0 + lun[i])
            spi.write(data[0])
            spi.write(data[1])
            spi.write(data[2])
        spi.write(0xff)
        spi.write(0xff)
        spi.write(0xff)
        spi.write(0xff)
```

（3）循环点亮灯环

参考代码:

```
c = [0, 45, 200]#定义一个颜色
while True:
    for q in range(12):
        a = [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]  # 让 12 个灯亮度为 2
        a[q] = 20  # 突出亮度的灯亮度设为 20
        on_led(a, c)  # 彩色灯环控制(亮度，颜色)
        print(a)
        time.sleep(0.02)  # 休眠 0.1 秒
```

五、操作流程

单击“ ▸▸ ”，运行此 jupyter lab 程序，可看到语音板麦克风阵列灯循环点亮，每次一个灯最亮。

机械臂认知和基础操作

一、实验目的

（1）了解机械臂系统硬件。

（2）了解机械臂基础操作；

二、实验内容

本实验要求机械臂系统硬件结构的基础上，掌握系统的硬件系统原理和端口资源分配，通过运行演示程序，了解机械臂的基础操作和基础控制。

三、实验环境

硬件环境	JetsonNX
操作系统	Linux
实验设备	机械臂，Jetson NX
实验配件	教具

四、实验原理

1.机械臂系统认知

机械臂系统硬件布局如图 1 所示，其包含部件如下：

图 1 机械臂系统硬件布局

2. 机械臂系统硬件参数

机械臂的硬件包含了 5 个总线舵机，1-4 轴使用的是 15KG 总线舵机，5 轴使用的是 35KG 总线舵机，舵机采用总线方式连接，舵机的参数表如表 1 所示。

表 1 机械臂参数表

轴	物理角度	软限位角度	工作电压范围	额定电压	空载电流	空载速度	额定扭矩	堵转扭矩（静态）	堵转电流	信号宽度范围	回中位角度误差
J1	0-300°	0-240°	6.0~7.4V	7.4V	≤300 mA	≤0.28 sec/60°	3.5 kgf.cm	≥15 kgf.cm	≤3.2 A	96~4000	≤1.0°
J2	0-300°	24-216°									
J3	0-300°	16-240°									
J4	0-300°	10-240°									
J5	0-320°	0-130°	6.0~8.4V	7.4V	≤200mA	≤0.16sec./60°	8.0kgf.cm	≥35kgf.cm	≤5.0A	0→4096	≤1.0°

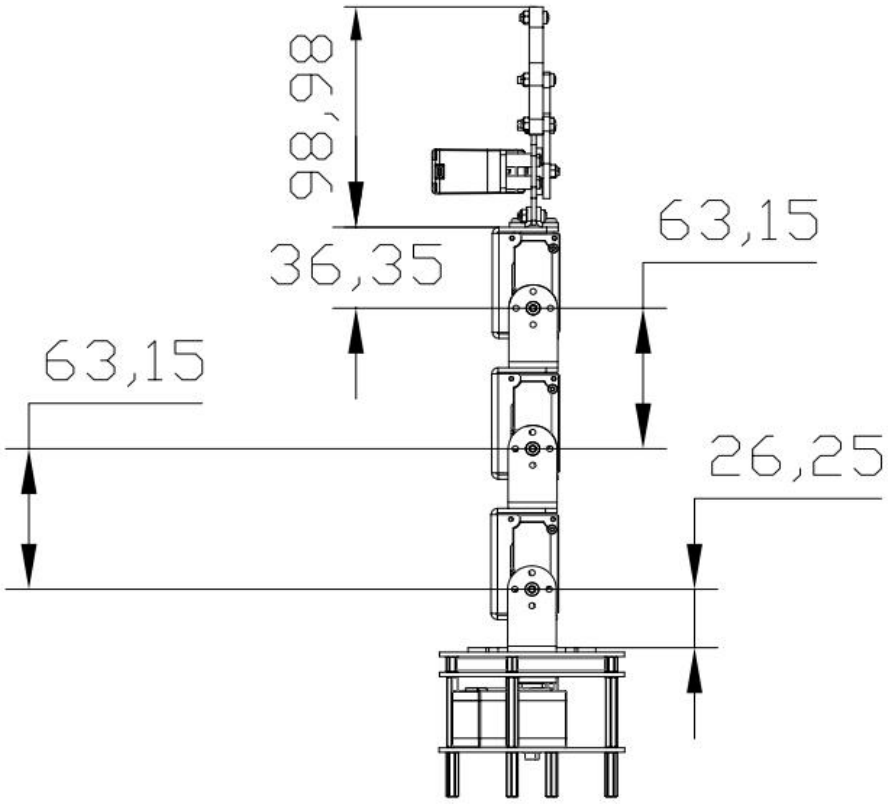


图 2 机械臂尺寸图

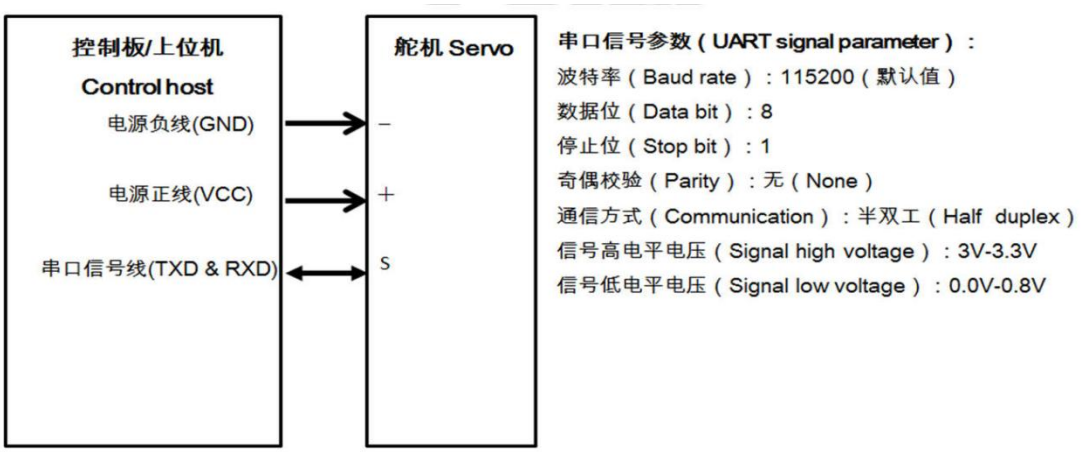


图 3 总线舵机电路及通讯

3. 机械臂相关控制函数

（1）bus_servo(*self*, idd, jioa, tim)

舵机控制（id，角度，时间）：此函数是对单个舵机角度进行控制，idd 是需要控制的舵机编号 1-5，jioa 是转动的角度（单位是：°），tim 是转动 jioa 角度的时间（单位是：ms），tim 值越小，速度越快。

（2）bus_servo_get(*self*, idd)

舵机读取;根据 idd 读取当前 idd 角度，角度是函数的返回值

（3）bus_servo_all(*self*, a, b, c, d, e, tim)

舵机同时控制（1，2，3，4，5，时间）:同时控制 1-5 关节同时运动，参数 a,b,c,d，e 是每个关节转动的角度，tim 是每个关节转动角度所给的时间。

（4）bus_servo_niuju_off(*self*, idd)

扭矩关闭（舵机编号）0xfe 为全部控制：idd 为舵机编号，控制机械臂不使能，机器人移动时必须是使能状态，在不使能的状态下可以手动移动机械臂。

（5）bus_servo_niuju_on(*self*, idd)

扭矩开启（舵机编号）0xfe 为全部控制：idd 为舵机编号，控制机械臂使能。

（6）bus_servo_pwr_on(*self*)

机械臂开启：机械臂在复位后必须要启动机械臂，启动后机械臂会移动到等待位。

（7）bus_pwr_off(*self*)

复位：使机器人复位，做完实验后需要让机器人复位。

※注意：不要随意更改机械臂控制函数里面的参数或代码，尤其是时间参数，该参数已经优化好，随意调参会导致机械臂异常，或者控制失败。

机械臂相关的控制函数可通过 Five_Robot_Control.py 程序文件查看。

五、实验步骤

机械臂采用总线方式进行控制，通过串口发送 16 进制数据控制舵机的转动，控制方式已经封装成机械臂控制类(Five_Robot_Control.py)，通过直接调用函数即可控制机械臂。

在开始做机械臂相关实验时，应先启动机械臂，通过按机械臂启动按钮，机械臂自动启动并运动到实验等待位，并释放扭矩(机械臂不使能，可以手动移动)。通过按复位按钮，机械臂会自动复位并运动到初始装配位置。

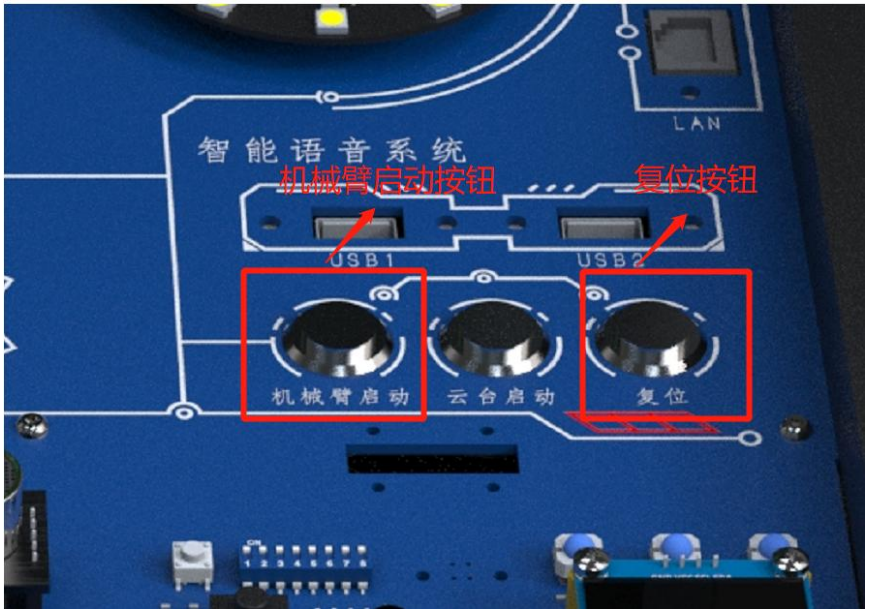


图 4 机械臂启动与复位按钮

机械臂的电路控制已经集成到底部的综合电源管理系统中，通过 JetsonNX 处理器发送指令给综合电源管理系统处理后转发给机械臂来控制机械臂的移动。

机器人控制移动需要先将机械臂使能后，机械臂才能运动。下面演示如何通过调用机器人控制类的函数来控制机械臂移动。

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）在单元格输入“%load 机械臂认知和基础操作.py”,点击运行按钮即可将程序载入到单元格。

2.导入库文件及函数

导入时间，机器人控制转换等相关库。

参考代码：

```
import time
from Five_Robot_Control import*
```

3.实例化机器人控制类对象

参考代码：

```
robot_control=Five_Robot_Control()
```

4.主程序：先使能机械臂，调用机械臂控制函数输入相应的度数中间保留一定时间，使机械臂有足够的时间移动到指定位置。

※注意：没有熟练操作之前，请不要大角度变化，快速移动机械臂，以防会发生撞机，损坏机械臂和实验箱器件。

参考代码：

```
if__name__==″__main__″:
robot_control. bus_servo_niuju_on(0xfe)#先使能
time. sleep(2)
robot_control. bus_servo_all(38, 45, 182, 219, 100, 1000)
time. sleep(2)
robot_control. bus_servo_all(38, 90, 182, 219, 0, 1000)
time. sleep(2)
robot_control. bus_servo_all(38, 45, 182, 219, 100, 1000)
time. sleep(2)
robot_control. bus_servo_all(113, 75, 182, 219, 0, 1000)
time. sleep(2)
robot_control. bus_servo_all(200, 75, 182, 219, 100, 1000)
time. sleep(2)
robot_control. bus_servo_all(113, 120, 167, 200, 0, 1000)
time. sleep(2)
robot_control. bus_servo_all(38, 45, 182, 219, 100, 1000)
```

机械臂示教和运动控制

一、实验目的

（1）了解机械臂运动控制方法。

（2）了解机械臂上位机逻辑及编程方式；

二、实验内容

本实验要求再对机械臂有一定的认知知识后，通过编写机械臂上位机控制界面，一般情况下，上位机是机械臂控制功能的集合，包含有机械臂控制完整功能，示例程序中机械臂控制功能有：启动，复位，归位，使能，轴运动控制，速度设置，步长设置，添加示教位置，位置编辑，程序打开与保存，拖动示教等功能。

机械臂上位机界面采用 PyQT5 进行编写，界面 UI 设计布局封装在“MainWindows.py”代码文件中，机械臂界面逻辑控制封装在“机械臂示教和运动控制.py”，机械臂的控制封装在“Five_Robot_Control.py”中，“MainWindows.ui”是使用 PyQT5 设计的 UI 界面文件，可以用 QtDesigner 打开，“MainWindows.py”可通过“MainWindows.ui”转化过来，在这里不多介绍，感兴趣的同学可以自行学习 PyQt 如何编写界面。

机械臂上位机操作说明：

1. 当机械臂处于复位状态时，点击启动按钮，机械臂会自动启动移动到等待位。

2. 启动可用两种选择，一种是使能机械臂，一种是拖动示教，使能机械臂后就不能操作拖动示教，同样的点击拖动示教后就不能使能，因为拖动示教状态下机械臂是不使能状态。

3. 如果使能机械臂，就可以通过轴运动控制按钮控制机械臂的移动，可以设置不同的速度，步长。

注意：不要过度操作，在不熟悉的情况下请慢，稳操作，以防发生撞机损坏机械臂。如果发生撞机，请将机械臂使能关闭，手动移动机械臂脱离撞机。

4. 如果点击拖动示教，机械臂自动进入拖动示教，这时机械臂是不使能状态，可以手动拖动机械臂到达需要位置，达到需要位置后，点按“机械臂启动”按钮一下即可将机械臂当前位置记录在示教界面，重复此操作即可完成机械拖动示教一套动作，完成后点击界面的运行按钮，机械臂即可按照拖动示教位置进行移动。

5. 在程序示教界面可以对程序进行添加，修改，更新位置等操作，可以修改每个单元格中的数值，且不要输入字母或者除数字以外的其他字符，否则机械臂会不执行该段位置直接跳过，或者报错。

三、实验环境

硬件环境	JetsonNX
操作系统	Linux
实验设备	机械臂，Jetson NX
实验配件	教具

四、实验原理

机械臂控制本质上是对机械臂舵机的控制，通过整合一些列的逻辑和功能实现机械臂上位机控制程序。无论是工业机械臂还是其他任何控制类的机械臂，其底层逻辑都是通过上层应用控制电机的转动。如何转动，转动多少都是有固定的数学算法解算，这就是机械臂的运动学，这在后面的实验中会具体的介绍讲解。

机械臂的上位机程序可以使用任何语言进行编写，Python，C，C++，C#，Java，Web 等都是可以的，根据自己的需要进行相应的程序设计，本实验采用的是 PyQt 本质上也是 Qt，Qt 的特性是可以跨平台，无论是 Linux 系统还是 Windows 系统都是可以使用的。

工业机械臂的原理也是一样，像 ABB 机器人上位机软件 RobotStudio，库卡机器人的 WorkVisual，安川机器人的 MotoPlus，发那科机器人的 ROBOGUIDE，这些软件都是每个厂家根据自己的机器人设计的一套通用的机器人开发上位机程序，这些上位机程序集成了，软件编程，虚拟仿真，机器人配置等，其本质也是基于机器人电机控制的基础上丰富完善功能，机器人电机的控制会进行封装，通过调用不用的函数实现不同的控制，本质上也是控制电机转动。电机控制又涉及到很多学科，像基本的知识简单的开环，闭环控制，PID 控制，电机的种类（步进，伺服，直流电机，交流电机，有刷，无刷等），机器人是一个复杂且庞大的

系统工程，是工业智慧的结晶。涉及到很多学科，且包含大量的数学知识，其底层的控制逻辑及算法本质上都是基于数学公式的计算实现。

机械臂上位机整个代码封装在“机械臂示教和运动控制.py”中，整个代码的结构如下图所示：

五、实验步骤

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）在单元格输入“%load 机械臂示教和运动控制. py”,点击运行按钮即可将程序载入到单元格。

2.导入库文件及函数

导入机器人控制，PyQt，线程，界面 UI 等相关库。

参考代码：

```
# region  库文件
import ctypes
import inspect
import re
import threading
import time
import tkinter
import numpy as np
from tkinter import filedialog#Linux 和 Windows 都可以使用
from PyQt5.QtCore import QRegExp
from PyQt5.QtGui import QStandardItem, QStandardItemModel, QRegExpValidator,
QIntValidator
from MainWindows import Ui_MainWindow
import sys
from PyQt5.QtWidgets import *
from Five_Robot_Control import *
# endregion
```

3.实例化机器人控制类对象

参考代码：

```
# region 机器人控制实例化对象
robot_control = Five_Robot_Control()
# endregion
```

4.全局变量

参考代码：

```
# region 全局变量
global item   # 保存 tableview 所有行数据
item = []
global isRunStop   # 程序运行是否停止标志位
isRunStop = True
# endregion
```

5.机器人控制业务逻辑

参考代码：

```
# region 机械臂示教和运动控制
class Five_Robot_Arm(QMainWindow, Ui_MainWindow):
    # region 构造函数
    def __init__(self):  # 创建构造函数
        super().__init__()  # 调用父类函数，继承
        self.setupUi(self)  # 调用 UI
        # 函数调用 初始化 tableView
        self.tableView_init()
        self.slot_init()  # 设置槽函数
        self.btn_ini(False)  # 控制相关按钮不使能
        self.btn_ini_1(False)  # 程序运行相关按钮不使能
        self.btn_enable_on.setEnabled(False)
        self.btn_enable_off.setEnabled(False)
        # 获取到 item 后开启发送线程
        self.thread_run_stop = threading.Thread(target=self.send_robot_data)
        self.thread_run_stop.start()
        # 设置背景显示文本
        self.lineEdit_file_name.setPlaceholderText("输入程序名称")
        #限制 line_Edit 的输入只能是数字
        self.lineEdit_recur.setValidator(QRegExpValidator(QRegExp("[0-9]+$")))#只能输入数
字
        self.lineEdit_step.setValidator(QIntValidator(1,240))#只能输入 1-240 数字
        self.lineEdit_speed.setValidator(QRegExpValidator(QRegExp("[0-9]+$")))#只能输入数
字


    # endregion
    # region 设置槽函数
    def slot_init(self):
        # 添加按钮
        self.btn_add.clicked.connect(self.btn_add_click)
        # 插入按钮
        self.btn_insert.clicked.connect(self.btn_insert_click)
        # 删除单行按钮
        self.btn_del.clicked.connect(self.btn_del_click)
        # 清除所有行按钮
        self.btn_clear.clicked.connect(self.btn_clear_click)
        # 上移按钮
        self.btn_move_up.clicked.connect(self.btn_move_up_click)
        # 下移按钮
        self.btn_move_down.clicked.connect(self.btn_move_down_click)
```

```
        # 复制按钮
        self.btn_copy.clicked.connect(self.btn_copy_click)
        # 粘贴按钮
        self.btn_stick.clicked.connect(self.btn_stick_click)
        # 运行程序按钮
        self.btn_run.clicked.connect(self.btn_run_click)
        # 运行停止按钮
        self.btn_stop.clicked.connect(self.btn_stop_click)
        # 单步运行按钮
        self.btn_step.clicked.connect(self.btn_step_click)
        # 更新按钮
        self.btn_renew.clicked.connect(self.btn_renew_click)
        # 复位按钮
        self.btn_reduce.clicked.connect(self.btn_reduce_click)
        # 归零按钮
        self.btn_zero.clicked.connect(self.btn_zero_click)
        # 启动按钮
        self.btn_start.clicked.connect(self.btn_start_click)
        # 关节 1：加,减
        self.btn_j1_add.clicked.connect(self.btn_j1_add_click)
        self.btn_j1_subtract.clicked.connect(self.btn_j1_subtract_click)
        # 关节 2：加,减
        self.btn_j2_add.clicked.connect(self.btn_j2_add_click)
        self.btn_j2_subtract.clicked.connect(self.btn_j2_subtract_click)
        # 关节 3：加,减
        self.btn_j3_add.clicked.connect(self.btn_j3_add_click)
        self.btn_j3_subtract.clicked.connect(self.btn_j3_subtract_click)
        # 关节 4：加,减
        self.btn_j4_add.clicked.connect(self.btn_j4_add_click)
        self.btn_j4_subtract.clicked.connect(self.btn_j4_subtract_click)
        # 夹爪开/关
        self.btn_clip_open.clicked.connect(self.btn_clip_open_click)
        self.btn_clip_close.clicked.connect(self.btn_clip_close_click)
        # 步长按钮
        self.btn_step_1.clicked.connect(self.btn_step_1_click)
        self.btn_step_5.clicked.connect(self.btn_step_5_click)
        self.btn_step_10.clicked.connect(self.btn_step_10_click)
        self.btn_step_15.clicked.connect(self.btn_step_15_click)
        # 速度按钮
        self.btn_speed_30.clicked.connect(self.btn_speed_30_click)
        self.btn_speed_100.clicked.connect(self.btn_speed_100_click)
        # 使能按钮
        self.btn_enable_on.clicked.connect(self.btn_enable_on_click)
        self.btn_enable_off.clicked.connect(self.btn_enable_off_click)
```

```
        # 程序保存
        self.btn_save_file.clicked.connect(self.btn_save_file_click)
        # 打开程序
        self.btn_open_file.clicked.connect(self.btn_open_file_click)
        # 拖动示教
        self.btn_teach_on.clicked.connect(self.btn_teach_on_click)
        self.btn_teach_off.clicked.connect(self.btn_teach_off_click)

    # endregion
    # region 启动按钮初始化
    # 控制部分按钮是否使能
    def btn_ini(self, bool):
        self.btn_teach_on.setEnabled(bool)
        self.btn_teach_off.setEnabled(bool)
        self.btn_zero.setEnabled(bool)
        self.btn_j1_add.setEnabled(bool)
        self.btn_j1_subtract.setEnabled(bool)
        self.btn_j2_add.setEnabled(bool)
        self.btn_j2_subtract.setEnabled(bool)
        self.btn_j3_add.setEnabled(bool)
        self.btn_j3_subtract.setEnabled(bool)
        self.btn_j4_add.setEnabled(bool)
        self.btn_j4_subtract.setEnabled(bool)
        self.btn_step_1.setEnabled(bool)
        self.btn_step_5.setEnabled(bool)
        self.btn_step_10.setEnabled(bool)
        self.btn_step_15.setEnabled(bool)
        self.lineEdit_step.setEnabled(bool)
        self.btn_speed_30.setEnabled(bool)
        self.btn_speed_100.setEnabled(bool)
        self.lineEdit_speed.setEnabled(bool)
        self.btn_clip_open.setEnabled(bool)
        self.btn_clip_close.setEnabled(bool)

    # 程序操作部分是否使能
    def btn_ini_1(self, bool):
        self.btn_open_file.setEnabled(bool)
        self.btn_save_file.setEnabled(bool)
        self.lineEdit_file_name.setEnabled(bool)
        self.lineEdit_recur.setEnabled(bool)
        self.btn_run.setEnabled(bool)
        self.btn_stop.setEnabled(bool)
        self.btn_step.setEnabled(bool)
        self.btn_add.setEnabled(bool)
```

```
        self.btn_insert.setEnabled(bool)
        self.btn_renew.setEnabled(bool)
        self.btn_move_up.setEnabled(bool)
        self.btn_move_down.setEnabled(bool)
        self.btn_stick.setEnabled(bool)
        self.btn_copy.setEnabled(bool)
        self.btn_del.setEnabled(bool)
        self.btn_clear.setEnabled(bool)

    # 程序操作部分是否使能排除停止按钮
    def btn_ini_2(self, bool):
        self.btn_open_file.setEnabled(bool)
        self.btn_save_file.setEnabled(bool)
        self.lineEdit_file_name.setEnabled(bool)
        self.lineEdit_recur.setEnabled(bool)
        self.btn_run.setEnabled(bool)
        self.btn_step.setEnabled(bool)
        self.btn_add.setEnabled(bool)
        self.btn_insert.setEnabled(bool)
        self.btn_renew.setEnabled(bool)
        self.btn_move_up.setEnabled(bool)
        self.btn_move_down.setEnabled(bool)
        self.btn_stick.setEnabled(bool)
        self.btn_copy.setEnabled(bool)
        self.btn_del.setEnabled(bool)
        self.btn_clear.setEnabled(bool)
        self.tableView.setEnabled(bool)

    # endregion
    # region 拖动示教
    def btn_teach_on_click(self):
        robot_control.bus_servo_niuju_off(0xfe)   # 关闭使能
        self.btn_teach_on.setEnabled(False)   # 点击拖动示教后开启线程就不能再点击
        self.btn_teach_off.setEnabled(True)
        self.btn_enable_on.setEnabled(False)
        self.btn_enable_off.setEnabled(False)
        self.btn_ini_1(True)
        self.btn_ini(False)#拖动示教打开，控制不能操作
        self.btn_teach_off.setEnabled(True)#拖动示教关闭可用
        self.btn_teach_on.setStyleSheet("QPushButton {text-align : center;\n"
                                        "                    background-color : green;\n"
                                        "                    font: bold;\n"
                                        "                    border-color: gray;\n"
                                        "                    border-width: 2px;\n"
```

```
                                                "                   border-radius: 0px;\n"
                                                "                   padding: 6px;\n"
                                                "                   height : 14px;\n"
                                                "                   border-style: outset;\n"
                                                "                   font : 14px;}\n")
        self.btn_teach_off.setStyleSheet("QPushButton {text-align : center;\n"
                                                "                   background-color :
rgb(216,216,216);\n"
                                                "                   font: bold;\n"
                                                "                   border-color: gray;\n"
                                                "                   border-width: 2px;\n"
                                                "                   border-radius: 0px;\n"
                                                "                   padding: 6px;\n"
                                                "                   height : 14px;\n"
                                                "                   border-style: outset;\n"
                                                "                   font : 14px;}\n")
        self.btn_enable_on.setStyleSheet("QPushButton {text-align : center;\n"
                                                "                   background-color :
rgb(216,216,216);\n"
                                                "                   font: bold;\n"
                                                "                   border-color: gray;\n"
                                                "                   border-width: 2px;\n"
                                                "                   border-radius: 15px;\n"
                                                "                   padding: 6px;\n"
                                                "                   height : 14px;\n"
                                                "                   border-style: outset;\n"
                                                "                   font : 14px;}\n")
        self.thread_receive = threading.Thread(target=self.receive_data)  # 接收拖动示教数
据线程
        self.thread_receive.start()

    def btn_teach_off_click(self):
        self.btn_enable_on.setEnabled(True)
        self.btn_enable_off.setEnabled(True)
        self.btn_teach_on.setEnabled(True)  # 与拖动示教开关相反
        self.btn_teach_off.setEnabled(False)
        self.btn_teach_on.setStyleSheet("QPushButton {text-align : center;\n"
                                                "                   background-color : rgb(216，
216，216);\n"
                                                "                   font: bold;\n"
                                                "                   border-color: gray;\n"
                                                "                   border-width: 2px;\n"
                                                "                   border-radius: 0px;\n"
                                                "                   padding: 6px;\n"
```

```
                                                "                   height : 14px;\n"
                                                "                   border-style: outset;\n"
                                                "                   font : 14px;}\n")
        self.btn_teach_off.setStyleSheet("QPushButton {text-align : center;\n"
                                                "                   background-color : red;\n"
                                                "                   font: bold;\n"
                                                "                   border-color: gray;\n"
                                                "                   border-width: 2px;\n"
                                                "                   border-radius: 0px;\n"
                                                "                   padding: 6px;\n"
                                                "                   height : 14px;\n"
                                                "                   border-style: outset;\n"
                                                "                   font : 14px;}\n")
        self.stop_thread(self.thread_receive)  # 关闭线程

    def receive_data(self):
        try:
            while True:
                time.sleep(0.1)
                nn = [0xFF, 0xFF, 0xFF, 0xFF, 0xFF]
                data_count = data_ser.inWaiting()
                if data_count:
                    data = str(binascii.b2a_hex(data_ser.read(data_count)))[2:-1]
                    nn[0] = int(data[10] + data[11], 16)  # 输入 16 进制的数并转换成 10
进制
                    nn[1] = int(data[12] + data[13], 16)  # 输入 16 进制的数并转换成 10
进制
                    nn[2] = int(data[14] + data[15], 16)  # 输入 16 进制的数并转换成 10
进制
                    nn[3] = int(data[16] + data[17], 16)  # 输入 16 进制的数并转换成 10
进制
                    nn[4] = int(data[18] + data[19], 16)  # 输入 16 进制的数并转换成 10
进制
                    if 0 <= nn[0] <= 240 and 24 <= nn[1] <= 216 and 16 <= nn[2] <= 240
and 10 <= nn[3] <= 240:
                        # 添加数据
                        item1 = QStandardItem(str(nn[0]))
                        item2 = QStandardItem(str(nn[1]))
                        item3 = QStandardItem(str(nn[2]))
                        item4 = QStandardItem(str(nn[3]))
                        item5 = QStandardItem(self.lineEdit_speed.text())
                        item6 = QStandardItem('0')
                        self.label_j1.setText(str(nn[0]))
                        self.label_j2.setText(str(nn[1]))
```

```
                            self.label_j3.setText(str(nn[2]))
                            self.label_j4.setText(str(nn[3]))
                            if 50 < nn[4] <= 250:
                                item7 = QStandardItem('0')
                            elif 0 <= nn[4] <= 50:
                                item7 = QStandardItem('1')
                            self.model.appendRow([item1, item2, item3, item4, item5, item6,
item7])
                    data_ser.flushInput()
            except Exception as e:
                print("数据接收线程：",e)

        # endregion
        # region 使能按钮
        def btn_enable_on_click(self):
            self.btn_ini(True)
            self.btn_ini_1(True)  # 添加程序后将所有程序相关操作按钮打开
            self.btn_teach_on.setStyleSheet("QPushButton {text-align : center;\n"
                                            "                background-color : rgb(216，
216，216);\n"
                                            "                font: bold;\n"
                                            "                border-color: gray;\n"
                                            "                border-width: 2px;\n"
                                            "                border-radius: 0px;\n"
                                            "                padding: 6px;\n"
                                            "                height : 14px;\n"
                                            "                border-style: outset;\n"
                                            "                font : 14px;}\n")
            self.btn_teach_on.setEnabled(False)  # 使能后不能运行拖动示教
            self.btn_teach_off.setEnabled(False)
            # self.btn_open_file.setEnabled(True)  # 打开使能后仅能加载程序和添加程序
            # self.btn_renew.setEnabled(True)
            # self.btn_add.setEnabled(True)
            robot_control.bus_servo_niuju_on(0xfe)
            degree = []
            degree = robot_control.bus_servo_get_all()
            if degree[0] <= 260:
                self.btn_enable_on.setStyleSheet("QPushButton {text-align : center;\n"
                                                 "                background-color :
green;\n"
                                                 "                font: bold;\n"
                                                 "                border-color: gray;\n"
                                                 "                border-width: 2px;\n"
                                                 "                border-radius: 15px;\n"
```

```
                                                "                padding: 6px;\n"
                                                "                height : 14px;\n"
                                                "                border-style: outset;\n"
                                                "                font : 14px;}\n")
            self.btn_enable_off.setStyleSheet("QPushButton {text-align : center;\n"
                                                "                background-color :
rgb(216,216,216);\n"
                                                "                font: bold;\n"
                                                "                border-color: gray;\n"
                                                "                border-width: 2px;\n"
                                                "                border-radius: 15px;\n"
                                                "                padding: 6px;\n"
                                                "                height : 14px;\n"
                                                "                border-style: outset;\n"
                                                "                font : 14px;}\n")
            self.label_j1.setText(str(degree[0]))
            self.label_j2.setText(str(degree[1]))
            self.label_j3.setText(str(degree[2]))
            self.label_j4.setText(str(degree[3]))

    def btn_enable_off_click(self):
        self.btn_ini(False)  # 使能关闭，所有的先关操作不使能
        self.btn_ini_1(False)
        self.btn_teach_on.setEnabled(True)
        self.btn_enable_on.setStyleSheet("QPushButton {text-align : center;\n"
                                            "                background-color : rgb(216，
216，216);\n"
                                            "                font: bold;\n"
                                            "                border-color: gray;\n"
                                            "                border-width: 2px;\n"
                                            "                border-radius: 15px;\n"
                                            "                padding: 6px;\n"
                                            "                height : 14px;\n"
                                            "                border-style: outset;\n"
                                            "                font : 14px;}\n")
        self.btn_enable_off.setStyleSheet("QPushButton {text-align : center;\n"
                                            "                background-color : red;\n"
                                            "                font: bold;\n"
                                            "                border-color: gray;\n"
                                            "                border-width: 2px;\n"
                                            "                border-radius: 15px;\n"
                                            "                padding: 6px;\n"
                                            "                height : 14px;\n"
                                            "                border-style: outset;\n"
```

```
                                            "                font : 14px;}\n")
        self.btn_teach_on.setStyleSheet("QPushButton {text-align : center;\n"
                                        "                background-color : rgb(216，
216，216);\n"
                                        "                font: bold;\n"
                                        "                border-color: gray;\n"
                                        "                border-width: 2px;\n"
                                        "                border-radius: 0px;\n"
                                        "                padding: 6px;\n"
                                        "                height : 14px;\n"
                                        "                border-style: outset;\n"
                                        "                font : 14px;}\n")
        robot_control.bus_servo_niuju_off(0xfe)

    # endregion
    # region 复位按钮点击事件
    def btn_reduce_click(self):
        self.btn_ini(False)
        self.btn_ini_1(False)
        self.btn_enable_on.setEnabled(False)
        self.btn_enable_off.setEnabled(False)
        robot_control.bus_pwr_off()
        self.btn_enable_on.setStyleSheet("QPushButton {text-align : center;\n"
                                         "                background-color : rgb(216，
216，216);\n"
                                         "                font: bold;\n"
                                         "                border-color: gray;\n"
                                         "                border-width: 2px;\n"
                                         "                border-radius: 15px;\n"
                                         "                padding: 6px;\n"
                                         "                height : 14px;\n"
                                         "                border-style: outset;\n"
                                         "                font : 14px;}\n")
        self.btn_enable_off.setStyleSheet("QPushButton {text-align : center;\n"
                                          "                background-color : red;\n"
                                          "                font: bold;\n"
                                          "                border-color: gray;\n"
                                          "                border-width: 2px;\n"
                                          "                border-radius: 15px;\n"
                                          "                padding: 6px;\n"
                                          "                height : 14px;\n"
                                          "                border-style: outset;\n"
                                          "                font : 14px;}\n")
        self.btn_teach_on.setStyleSheet("QPushButton {text-align : center;\n"
```

```
                                            "                   background-color : rgb(216，
216，216);\n"
                                            "                   font: bold;\n"
                                            "                   border-color: gray;\n"
                                            "                   border-width: 2px;\n"
                                            "                   border-radius: 0px;\n"
                                            "                   padding: 6px;\n"
                                            "                   height : 14px;\n"
                                            "                   border-style: outset;\n"
                                            "                   font : 14px;}\n")
        self.btn_teach_off.setStyleSheet("QPushButton {text-align : center;\n"
                                            "                     background-color : rgb(216，
216，216);\n"
                                            "                     font: bold;\n"
                                            "                     border-color: gray;\n"
                                            "                     border-width: 2px;\n"
                                            "                     border-radius: 0px;\n"
                                            "                     padding: 6px;\n"
                                            "                     height : 14px;\n"
                                            "                     border-style: outset;\n"
                                            "                     font : 14px;}\n")

    # endregion
    # region 运行按钮点击事件
    def btn_run_click(self):
        robot_control.bus_servo_niuju_on(0xfe)  # 使能打开
        self.btn_teach_on.setEnabled(False)
        self.btn_teach_off.setEnabled(False)
        self.btn_enable_on.setStyleSheet("QPushButton {text-align : center;\n"
                                            "                   background-color : green;\n"
                                            "                   font: bold;\n"
                                            "                   border-color: gray;\n"
                                            "                   border-width: 2px;\n"
                                            "                   border-radius: 15px;\n"
                                            "                   padding: 6px;\n"
                                            "                   height : 14px;\n"
                                            "                   border-style: outset;\n"
                                            "                   font : 14px;}\n")
        self.btn_enable_off.setStyleSheet("QPushButton {text-align : center;\n"
                                            "                     background-color : rgb(216，
216，216);\n"
                                            "                     font: bold;\n"
                                            "                     border-color: gray;\n"
                                            "                     border-width: 2px;\n"
```

```
                                                "                    border-radius: 15px;\n"
                                                "                    padding: 6px;\n"
                                                "                    height : 14px;\n"
                                                "                    border-style: outset;\n"
                                                "                    font : 14px;}\n")
            self.btn_teach_on.setStyleSheet("QPushButton {text-align : center;\n"
                                                "                    background-color : rgb(216，
216，216);\n"
                                                "                    font: bold;\n"
                                                "                    border-color: gray;\n"
                                                "                    border-width: 2px;\n"
                                                "                    border-radius: 15px;\n"
                                                "                    padding: 6px;\n"
                                                "                    height : 14px;\n"
                                                "                    border-style: outset;\n"
                                                "                    font : 14px;}\n")
            self.btn_ini_2(False)  # 程序运行过程中其他按钮不使能，除了停止外
            self.btn_ini(False)  # 控制部分不使能
            try:
                rowCount = self.model.rowCount()
                colCount = self.model.columnCount()
                global item
                test = []
                for i in range(0, rowCount):
                    # self.tableView.setSelectionBehavior(QAbstractItemView.SelectRows)
                    # self.tableView.setStyleSheet("selection-background-color:rgb(0,150,200)")
                    # self.tableView.selectRow(i)
                    for j in range(0, colCount):
                        test.append(self.model.index(i, j).data())
                    item.append(test)
                    test = []  # 清除速度，不累加
            except Exception as e:
                print(e)

        def send_robot_data(self):
            try:
                while True:
                    time.sleep(0.1)
                    value = self.lineEdit_recur.text()
                    if value == "":
                        value = '1'
                    for t in range(0, int(value)):
                        global item
                        for i in range(0, len(item)):
```

```
                                # 判断表格数据是否是数值类型，防止发送数据线程报错
                                if self.is_number(item[i][0]) == True and self.is_number(item[i][1])
== True and self.is_number(
                                        item[i][2]) == True and self.is_number(item[i][3]) == True
and self.is_number(
                                        item[i][4]) == True and self.is_number(item[i][5]) == True
and self.is_number(
                                        item[i][6]) == True:
                                    if isRunStop:
                                        if int(item[i][6]) == 0:
                                            clip_value = 100    # 夹爪关
                                        else:
                                            clip_value = 0    # 夹爪开
                                        if int(item[i][5]) >= 500:
                                            time.sleep(int(item[i][5]) / 1000)
                                        else:
                                            time.sleep(0.5)
                                        if 0<=int(item[i][0])<=240 and 24<=int(item[i][1])<=216
and 16<=int(item[i][2])<=240 and 10<=int(item[i][3])<=240:
                                            robot_control.bus_servo_all(int(item[i][0]),
int(item[i][1]), int(item[i][2]),
                                                                        int(item[i][3]),
                                                                        clip_value,
int(item[i][4]))
                                            self.label_j1.setText(item[i][0])
                                            self.label_j2.setText(item[i][1])
                                            self.label_j3.setText(item[i][2])
                                            self.label_j4.setText(item[i][3])
                                            self.lineEdit_recur.setText(str(int(value) - t))#程序执
行次数递减
                        item = []
            except Exception as e:
                print("数据发送线程：",e)

    def btn_stop_click(self):

        global isRunStop
        if isRunStop == True:
            isRunStop = False
            self.btn_stop.setText("开始")
            self.btn_run.setEnabled(False)
        else:
            isRunStop = True
            self.btn_stop.setText("停止")
```

```
                self.btn_ini_2(True)
                self.btn_ini(True)
            print(isRunStop)

        # endregion
        # region 单步运行按钮事件
        def btn_step_click(self):
            try:
                index = self.tableView.currentIndex()  # 取得当前选中行的 index
                row = index.row()  # 当前行
                colCount = self.model.columnCount()  # 所有列
                test = []
                for i in range(0, colCount):
                    test.append(self.model.index(row, i).data())
                # 判断单元格是否是数值类型切是否在角度范围内
                if self.is_number(test[0]) == True and self.is_number(test[1]) == True and
self.is_number(
                    test[2]) == True and self.is_number(test[3]) == True and self.is_number(
                    test[4]) == True and self.is_number(test[5]) == True and
self.is_number(test[6]) == True:
                    if int(test[6]) == 0:
                        clip_value = 100  # 夹爪关
                    else:
                        clip_value = 0  # 夹爪开
                    if int(test[5]) >= 500:
                        time.sleep(int(test[5]) / 1000)
                    else:
                        time.sleep(0.5)
                    if 0 <= int(test[0]) <= 240 and 24 <= int(test[1]) <= 216 and 16 <= int(test[2])
<= 240 and 10 <= int(test[3]) <= 240:
                        robot_control.bus_servo_all(int(test[0]), int(test[1]), int(test[2]),
int(test[3]), clip_value,
                                                    int(test[4]))
                        self.label_j1.setText(test[0])
                        self.label_j2.setText(test[1])
                        self.label_j3.setText(test[2])
                        self.label_j4.setText(test[3])
                        test = []
            except Exception as e:
                print(e)

        # endregion
        # region 更新按钮事件
        def btn_renew_click(self):
```

```
            index = self.tableView.currentIndex()   # 取得当前选中行的 index
            if index != None:
                self.model.setItem(index.row(), 0, QStandardItem(self.label_j1.text()))
                self.model.setItem(index.row(), 1, QStandardItem(self.label_j2.text()))
                self.model.setItem(index.row(), 2, QStandardItem(self.label_j3.text()))
                self.model.setItem(index.row(), 3, QStandardItem(self.label_j4.text()))
                self.model.setItem(index.row(), 4, QStandardItem(self.lineEdit_speed.text()))

        # endregion
        # region 归零按钮事件
        def btn_zero_click(self):
            wait_point = [38, 45, 182, 219, 0, 1000]
            robot_control.bus_servo_all(wait_point[0], wait_point[1], wait_point[2], wait_point[3],
wait_point[4],
                                        wait_point[5])
            self.label_j1.setText(str(wait_point[0]))
            self.label_j2.setText(str(wait_point[1]))
            self.label_j3.setText(str(wait_point[2]))
            self.label_j4.setText(str(wait_point[3]))

        # endregion
        # region 启动按钮
        def btn_start_click(self):
            robot_control.bus_servo_pwr_on()
            self.btn_enable_on.setEnabled(True)
            self.btn_enable_off.setEnabled(True)
            self.btn_teach_on.setEnabled(True)
            # self.btn_enable_on.setStyleSheet("QPushButton {text-align : center;\n"
            #                                  "                 background-color :
rgb(216，216，216);\n"
            #                                  "                 font: bold;\n"
            #                                  "                 border-color: gray;\n"
            #                                  "                 border-width: 2px;\n"
            #                                  "                 border-radius: 15px;\n"
            #                                  "                 padding: 6px;\n"
            #                                  "                 height : 14px;\n"
            #                                  "                 border-style: outset;\n"
            #                                  "                 font : 14px;}\n")
            # self.btn_enable_off.setStyleSheet("QPushButton {text-align : center;\n"
            #                                   "                 background-color : red;\n"
            #                                   "                 font: bold;\n"
            #                                   "                 border-color: gray;\n"
            #                                   "                 border-width: 2px;\n"
            #                                   "                 border-radius: 15px;\n"
```

```
            #                                       "                   padding: 6px;\n"
            #                                       "                   height : 14px;\n"
            #                                       "                   border-style: outset;\n"
            #                                       "                   font : 14px;}\n")

    # endregion
    # region 单元格相关操作
    def tableView_init(self):
        # self.tableView.setAlternatingRowColors(True)
        # 4 行 3 列
        self.model = QStandardItemModel(0, 0)
        # 设置表头
        self.model.setHorizontalHeaderLabels(
            ['关节 1(0-240°)', '关节 2(24-216°)', '关节 3(16-240°)', '关节 4(10-240°)', '速度(ms/deg)', '延迟(ms)', '夹爪(开/关)'])
        # 列宽自适应充满表格
        self.tableView.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)  # 所有列自动拉伸，充满界面
        # self.tableView.setSelectionMode(QAbstractItemView.SingleSelection)  # 设置只能单个
        self.tableView.setSelectionBehavior(QAbstractItemView.SelectRows)  # 设置只能选中整行
        # 关联 QTableView 控件和 Model
        self.tableView.setModel(self.model)

        layout = QVBoxLayout()
        layout.addWidget(self.tableView)
        self.setLayout(layout)
    # 下移行
    def btn_move_down_click(self):
        index = self.tableView.currentIndex()  # 取得当前选中行的 index
        row = index.row()
        if row < self.model.rowCount() - 1:
            self.model.insertRow(row + 2)
            for i in range(self.model.columnCount()):
                self.model.setItem(row + 2, i, self.model.takeItem(row, i))
            self.model.removeRow(row)

    # 上移行
    def btn_move_up_click(self):
        index = self.tableView.currentIndex()  # 取得当前选中行的 index
        row = index.row()
        if row > 0:
            self.model.insertRow(row - 1)
```

```
                for i in range(self.model.columnCount()):
                    self.model.setItem(row - 1, i, self.model.takeItem(row + 1, i))
                self.model.removeRow(row + 1)

    # 添加行
    def btn_add_click(self):
        # 添加数据
        item1 = QStandardItem(self.label_j1.text())
        item2 = QStandardItem(self.label_j2.text())
        item3 = QStandardItem(self.label_j3.text())
        item4 = QStandardItem(self.label_j4.text())
        item5 = QStandardItem(self.lineEdit_speed.text())
        item6 = QStandardItem('0')
        item7 = QStandardItem('1')
        self.model.appendRow([item1, item2, item3, item4, item5, item6, item7])
    #插入行
    def btn_insert_click(self):
        # 插入数据
        item1 = QStandardItem(self.label_j1.text())
        item2 = QStandardItem(self.label_j2.text())
        item3 = QStandardItem(self.label_j3.text())
        item4 = QStandardItem(self.label_j4.text())
        item5 = QStandardItem(self.lineEdit_speed.text())
        item6 = QStandardItem('0')
        item7 = QStandardItem('1')
        index = self.tableView.currentIndex()  # 取得当前选中行的 index
        row = index.row()
        self.model.insertRow(row + 1, [item1, item2, item3, item4, item5, item6, item7])  # 当
前行插入数据
    # 删除行
    def btn_del_click(self):
        index = self.tableView.currentIndex()  # 取得当前选中行的 index
        self.model.removeRow(index.row())  # 通过 index 的 row()操作得到行数进行删除

    # 清除所有行
    def btn_clear_click(self):
        # 会全部清空，包括那个标准表头
        self.model.clear()
        # 设置表头
        self.model.setHorizontalHeaderLabels(
            ['关节 1(0-240°)', '关节 2(24-216°)', '关节 3(16-240°)', '关节 4(10-240°)', '速度
(ms/deg)', '延迟(ms)', '夹爪(开/关)'])

    # 复制
```

```
    def btn_copy_click(self):
        text = self.selected_tb_text()  # 获取当前表格选中的数据
        if text:
            clipboard = QApplication.clipboard()
            clipboard.setText(text)
            # pyperclip.copy(text) #  复制数据到粘贴板

    # 粘贴
    def btn_stick_click(self):
        self.paste_tb_text()

    # 复制单元格文本
    def paste_tb_text(self):
        try:
            indexes = self.tableView.selectedIndexes()  # 获取表格对象中被选中的数据索
引列表
            for index in indexes:
                index = index
                break
            r, c = index.row(), index.column()
            text = QApplication.clipboard().text()
            ls = text.split('\n')
            ls1 = []
            for row in ls:
                ls1.append(row.split('\t'))
            model = self.tableView.model()
            rows = len(ls)
            columns = len(ls1[0])
            for row in range(rows):
                for column in range(columns):
                    item = QStandardItem()
                    item.setText((str(ls1[row][column])))
                    model.setItem(row + r, column + c, item)
        except Exception as e:
            print(e)
            return

    # 选择单元格文本
    def selected_tb_text(self):
        try:
            indexes = self.tableView.selectedIndexes()  # 获取表格对象中被选中的数据索
引列表
            indexes_dict = {}
            for index in indexes:  # 遍历每个单元格
```

```
                row, column = index.row(), index.column()  # 获取单元格的行号，列号
                if row in indexes_dict.keys():
                    indexes_dict[row].append(column)
                else:
                    indexes_dict[row] = [column]

            # 将数据表数据用制表符(\t)和换行符(\n)连接，使其可以复制到 excel 文件中
            text = ''
            for row, columns in indexes_dict.items():
                row_data = ''
                for column in columns:
                    data = self.tableView.model().item(row, column).text()
                    if row_data:
                        row_data = row_data + '\t' + data
                    else:
                        row_data = data

                if text:
                    text = text + '\n' + row_data
                else:
                    text = row_data
            return text
        except BaseException as e:
            print(e)
            return ''

# endregion
# region  关节控制
def btn_j1_add_click(self):
    degree = int(self.label_j1.text()) + int(self.lineEdit_step.text())
    if degree <= 240:
        robot_control.bus_servo(1, degree, int(self.lineEdit_speed.text()))
        self.label_j1.setText(str(degree))

def btn_j1_subtract_click(self):
    degree = int(self.label_j1.text()) - int(self.lineEdit_step.text())
    if degree >= 0:
        robot_control.bus_servo(1, degree, int(self.lineEdit_speed.text()))
        self.label_j1.setText(str(degree))

def btn_j2_add_click(self):
    degree = int(self.label_j2.text()) + int(self.lineEdit_step.text())
    if degree <= 216:
        robot_control.bus_servo(2, degree, int(self.lineEdit_speed.text()))
```

```
            self.label_j2.setText(str(degree))

    def btn_j2_subtract_click(self):
        degree = int(self.label_j2.text()) - int(self.lineEdit_step.text())
        if degree >= 24:
            robot_control.bus_servo(2, degree, int(self.lineEdit_speed.text()))
            self.label_j2.setText(str(degree))

    def btn_j3_add_click(self):
        degree = int(self.label_j3.text()) + int(self.lineEdit_step.text())
        if degree <= 240:
            robot_control.bus_servo(3, degree, int(self.lineEdit_speed.text()))
            self.label_j3.setText(str(degree))

    def btn_j3_subtract_click(self):
        degree = int(self.label_j3.text()) - int(self.lineEdit_step.text())
        if degree >= 16:
            robot_control.bus_servo(3, degree, int(self.lineEdit_speed.text()))
            self.label_j3.setText(str(degree))

    def btn_j4_add_click(self):
        degree = int(self.label_j4.text()) + int(self.lineEdit_step.text())
        if degree <= 240:
            robot_control.bus_servo(4, degree, int(self.lineEdit_speed.text()))
            self.label_j4.setText(str(degree))

    def btn_j4_subtract_click(self):
        degree = int(self.label_j4.text()) - int(self.lineEdit_step.text())
        if degree >= 10:
            robot_control.bus_servo(4, degree, int(self.lineEdit_speed.text()))
            self.label_j4.setText(str(degree))

    def btn_clip_open_click(self):
        robot_control.bus_servo(5, 0, int(self.lineEdit_speed.text()))

    def btn_clip_close_click(self):
        robot_control.bus_servo(5, 100, int(self.lineEdit_speed.text()))

    def btn_step_1_click(self):
        self.lineEdit_step.setText('1')

    def btn_step_5_click(self):
        self.lineEdit_step.setText('5')
```

```
    def btn_step_10_click(self):
        self.lineEdit_step.setText('10')

    def btn_step_15_click(self):
        self.lineEdit_step.setText('15')

    def btn_speed_30_click(self):
        self.lineEdit_speed.setText('30')

    def btn_speed_100_click(self):
        self.lineEdit_speed.setText('100')

    # endregion
    # region 程序保存
    def btn_save_file_click(self):
        try:
            rowCount = self.model.rowCount()
            colCount = self.model.columnCount()
            test = []
            # 创建一个 txt 文件，文件名为 first file,并向文件写入 msg
            now_time = time.strftime('%Y-%m-%d-%H-%M-%S-%M',
time.localtime(time.time()))  # 获取系统时间
            program_name = self.lineEdit_file_name.text()
            if program_name == "":  # 判断程序文件名是否为空，如果为空就以当前时间
保存为文件名
                program_name = now_time
            file = open('ProgramFile/program_' + str(program_name) + '.txt', 'w').close()  #
清空程序文件
            for i in range(0, rowCount):  # 一行一行写入 txt,方便 txt 读取
                for j in range(0, colCount):
                    test.append(self.model.index(i, j).data())
                with open('ProgramFile/program_' + str(program_name) + '.txt', "a",
encoding='utf-8') as f:  # 追加写入
                    f.write(str(test))
                    f.write('\n')
                test = []  # 清除数据，不累加
        except Exception as e:
            print(e)

    # endregion
    # region 打开程序
    def btn_open_file_click(self):
        try:
            root = tkinter.Tk()
```

```
                root.withdraw()
                filename = filedialog.askopenfilename(initialdir = 'ProgramFile')
                list = []
                with open(filename, 'r') as f:
                    self.btn_clear_click()  # 清除所有行，再写入
                    my_data = f.readlines()  # txt 中所有字符串读入 my_data，得到的是一个
list
                    for line in my_data:  # 将文本字符串转成数组字符串
                        line = line.strip()  # 移除字符串头尾
                        line = line.strip("[]")  # 移除字符串头尾[]
                        line_data = line.split(',')  # 按照逗号分隔
                        line = line.strip()  # 移除字符串头尾引号
                        list.append(line_data)  # 添加到数组
                    all_list = []  # 字符串数组转为整型后所有行的数组
                    single_list = []  # 保存每一行转换的数组
                    for i in range(0, len(list)):  # 将数组字符串转数组整型，eval 去掉引号并
转为整型：‘1’→1
                        for j in range(0, len(list[i])):
                            value = int(eval(list[i][j]))
                            single_list.append(value)
                        all_list.append(single_list)
                        single_list = []
                    for i in range(0, len(all_list)):
                        for j in range(0, len(all_list[i])):
                            self.model.setItem(i, j, QStandardItem(str(all_list[i][j])))  # 将数
据填充到每一个单元格
            except Exception as e:
                print(e)

    # endregion
    #region 判断字符串是否是数值类型
    def is_number(self,s):
        try:
            int(s)
            return True
        except Exception as e:
            print("数值转换整型错误:",e)
            pass
        try:
            import unicodedata
            unicodedata.numeric(s)
        except Exception as e:
            print("数值转换整型错误:", e)
        return False
```

```
        #endregion
        # region 退出程序，关闭线程
        def closeEvent(self, event):
            if self.thread_run_stop.is_alive():
                self.stop_thread(self.thread_run_stop)  # 关闭线程
            if self.thread_receive.is_alive():
                self.stop_thread(self.thread_receive)  # 关闭线程
            self.close()

        def _async_raise(self, tid, exctype):
            """raises the exception, performs cleanup if needed"""
            tid = ctypes.c_long(tid)
            if not inspect.isclass(exctype):
                exctype = type(exctype)
            res = ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, ctypes.py_object(exctype))
            if res == 0:
                raise ValueError("invalid thread id")
            elif res != 1:
                # """if it returns a number greater than one, you're in trouble,
                # and you should call it again with exc=NULL to revert the effect"""
                ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)
                raise SystemError("PyThreadState_SetAsyncExc failed")

        def stop_thread(self, thread):
            self._async_raise(thread.ident, SystemExit)
        # endregion
# region 主程序运行
if __name__ == "__main__":
        app = QApplication(sys.argv)
        five_robot = Five_Robot_Arm()
        five_robot.show()
        sys.exit(app.exec_())
# endregion
# endregion
```

机械臂与视觉系统标定

一、实验目的

（1）了解视觉标定原理与操作方法。

（2）了解机械臂世界坐标系与图像坐标系之间的关系转换；

二、实验内容

机械臂与视觉系统标定主要学习机械臂与视觉标定的原理及方法，弄懂世界坐标系，基座标系，图像坐标系之间的转换关系，弄懂如何从图像坐标计算世界坐标等，并能推导相应的公式，并将公式通过代码实现。

三、实验环境

硬件环境	JetsonNX
操作系统	Linux
实验设备	机械臂，Jetson NX
实验配件	教具

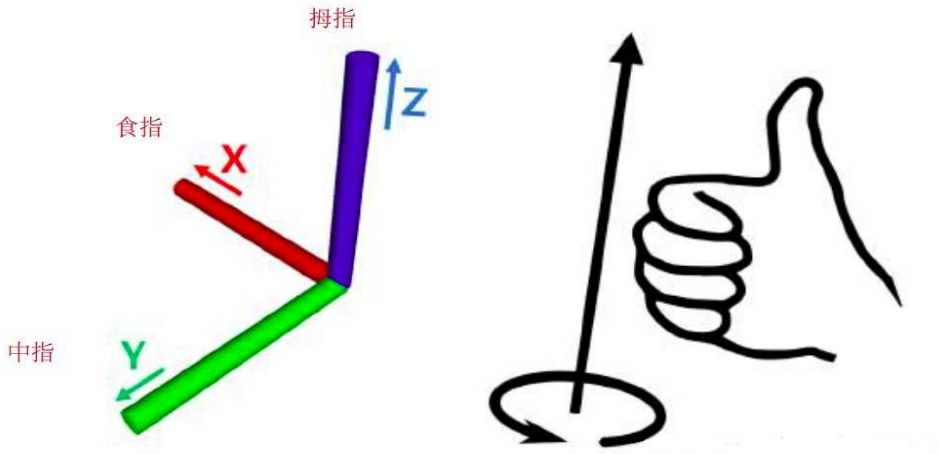
四、实验原理

前面已经学习了机械臂的控制方法，本节实验主要讲解视觉与机械臂之间的关系。我们知道机械臂要抓取一个物体，需要先知道物体的位置在哪里，然后通过机械臂示教方式定点抓取物体，假如不知道物体的位置在哪里，机械臂就无法抓取。那么我们所说的这个位置是在空间坐标中，物体和机械臂的相对坐标关系，我们把这个关系称作为世界坐标系也叫作笛卡尔坐标系。那么世界坐标关系可以描述为：在笛卡尔坐标系中，以原点为参考点，机械臂与物体的相对关系称作为机械臂的世界坐标系。同理，如果以机械臂的底座中点为笛卡尔坐标系的原点建立空间坐标系，那么物体与机械臂的相对关系我们称作为机械臂基座标系。理解了机械的世界坐标系和基座标系后，还有工件坐标系和工具坐标系，工件和工具坐标系主要是在工业型机械臂中会经常使用到，本机械臂不涉及，但如果需要理解透彻，就需要学习这两种坐标系。

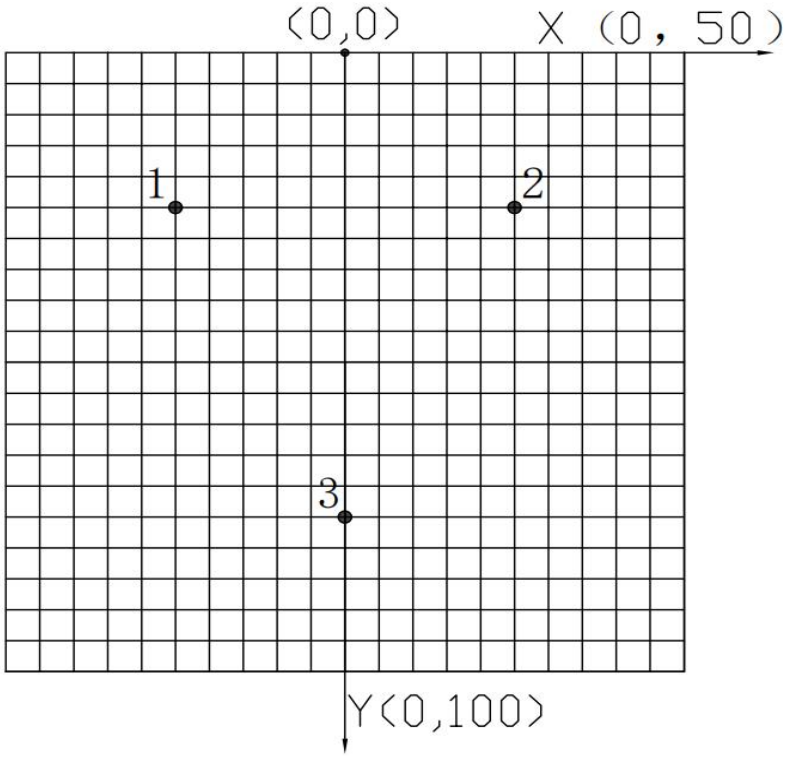
通常我们在建立了机械臂坐标系后，需要知道坐标系的各个坐标轴的关系，怎么确定这个关系呢？工业中常用的确定机械臂方法叫做“右手定则”，也有使用“左手定则”，根据自己的理解选择合适的方法，这里讲解“右手定则”：

机器人中的旋转轴使用 x，y 和 z 轴。正面是 x 轴的正方向，轴是红色（Red）。左边是 y 轴的正方向，轴用绿色（Green）表示。最后，上方是 z 轴的正方向，轴用蓝色（Blue）表示。为了便于记忆，您可以将 x 轴视为食指，将 y 轴视为中指，将 z 轴视为拇指。顺序是 x、y、z，且颜色是 RGB 颜色顺序。机器人的旋

转方向是右手定则，用右手卷住的方向是正（+）方向。



机械臂的坐标系我们清楚之后，就要知道图像的坐标系，我们把物体在图像中的位置叫做物体的像素坐标。也就是物体的中心点在图像中的像素值。
那么机械臂怎么知道物体的世界坐标关系呢？我们以机械臂的底座中点为世界坐标系的原点建立空间坐标系，通过给空间坐标系划定尺度信息就知道物体相对于基座标的（x, y）如下图所示：



我们把（0, 0）看作为是机械臂的基点坐标来建立一个平面坐标系（如果基点不在（0, 0）, 只需要算上机械臂底座的中点到（0, 0）的偏差值即可，本机械臂的偏差值在运动学正逆解代码中以 P 表示），平面坐标系的大小为 10*10cm 的方

格，从坐标系的角度来看 1, 2, 3 个点的坐标分别是（−2. 5, 2. 5）, (2. 5, 2. 5), (0, 7. 5)（注意：如果实际应用中应考虑视觉的安装方向，标定的时候世界坐标也与图像坐标方向一致），假如 1，2, 3 个点是机械臂需要抓取的物体，那么只需要将机械臂移动到该点即可，但是我们知道机械臂到达该点的（X, Y）坐标却不知到机械臂需要转动多少角度呢？这就涉及到机器人的运动学，我们可以通过机器人的运动学逆解出来机械臂需要转动的角度。逆解的代码在“Five_Robot_kinematics. py”，通过逆解我们知道世界坐标系到关节坐标系的转换关系，通过正解我们知道关节坐标系到世界坐标系的转换关系。该部分涉及到比较复杂的数学公式推导和机器人运动学导论的知识，同学们可以根据自己的兴趣了解该部分的内容，这里只讲解实现的具体方法，不做复杂的公式推导。

以上我们知道了物体与机械臂世界坐标系的关系，也知道物体的图像坐标系的关系，那么只需要解算出图像坐标与世界坐标的关系就可以通过知道图像坐标来计算出世界坐标，这个过程就叫做机械臂与视觉系统的标定。

我们知道了 1, 2, 3 个点的世界坐标关系分别是（point1_X, point1_Y），（point2_X, point2_Y），（point3_X, point3_Y），也知道了 1, 2, 3 像素坐标（pixel1_x, pixel1_y），（pixel2_x, pixel2_y），（pixel3_x, pixel3_y）
我们把图像坐标做是矩阵 A，世界坐标看做是矩阵 B，可以得到一个关系 $A*X=B$, 那么 $X=B*A^{-1}$, X 就叫做关系矩阵。我们通过三点标定的方式解出关系矩阵后，当通过视觉识别知道物体的像素坐标乘关系矩阵就可以求得世界坐标，机械臂就可以抓取物体。除了三点标定外，还可以通过九点标定，12 点标定等，原理都是想通的，九点标定的目的是去除耦合解，精度更高，但在一般的使用中三点标定的精度足够，有兴趣的同学可以试下 9 点标定，关于标定方面的知识，同学们可以看一下张正友标定，是目前很经典的标定方法，网上资料很多，这里不过多赘述。

标定的矩阵解算通过 opencv 里面的函数即可，M 即为关系矩阵：

```
#坐标转为数组
pts1=np. float32([[pixel1_x, pixel1_y], [pixel2_x, pixel2_y], [pixel3_x, pixel3_y]])
pts2=np. float32([[point1_X, point1_Y], [point2_X, point2_Y], [point3_X, point3_Y]])
M=cv2. getAffineTransform(pts1, pts2)#仿射变化，仿射矩阵为 2*3
```

五、实验步骤

1.运行 jupyter lab

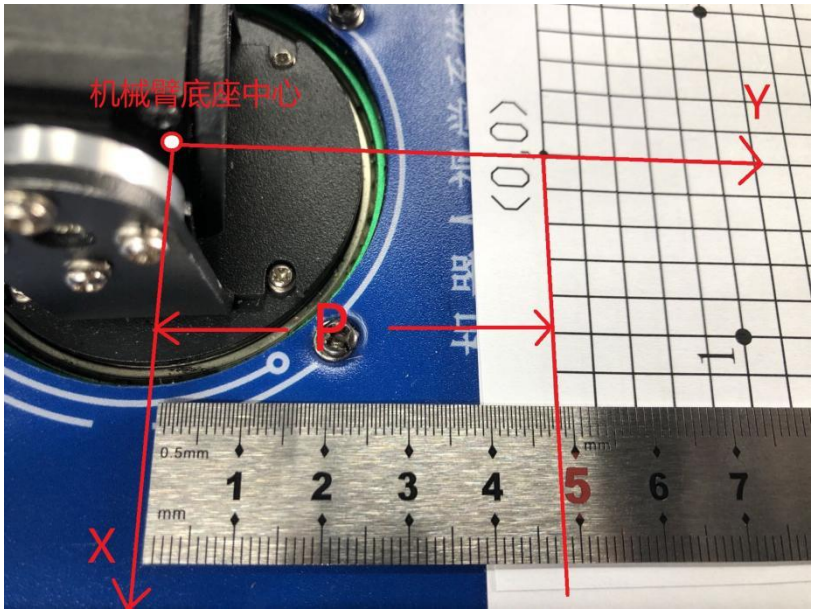
（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）在单元格输入“%load 机械臂示教与视觉系统标定. py”,点击运行按钮即可将程序载入到单元格（先不要运行程序）。

（5）将文件夹中的标定板用 A4 纸打印进行裁剪，裁剪好后，放在相机视野下面的白色区域，标定板坐标（0,0）对准机械臂中轴线方向也就是机械臂中轴线需要与 Y 轴重合，这样做的目的是减小计算。然后用直尺测量出机械臂基座中点到标定板 X 轴的的距离 P 填写在“Five_Robot_kinematics. py”参数中 P 值保存(按照实际的白色抓取区域，P 值为 4.3cm 左右，P 值是根据世界坐标区域位置改变的)，如图所示：



Five_Robot_kinematics.py

```
import math
import time
#region机械臂参数
#机械臂4连杆长度定义，单位厘米
P = 4.8
A1 = 2.6
A2 = 6.3
A3 = 6.3
A4 = 15
MAX_LEN = A2+A3+A4
MAX_HIGH = A1+A2+A3+A4
#endregion
机械臂正逆解
```

（6）打开相机采图软件采集一张相机标定板的图片，将图片拷贝保存在该实验文件夹中的 pic,并将图像重命名“1.jpg”。

2.导入库文件及函数

导入 OpenCV，numpy 等相关库。

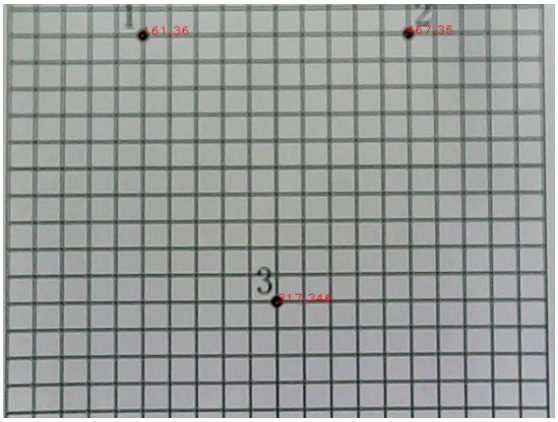
参考代码：

```
import cv2
import numpy as np
```

3.读入图片，对准图像中的 1,2,3 坐标点击鼠标左键，会自动将当前 1,2,3 的像素坐标记录图像中，同时结果图像会保存到 pic 文件夹中。

参考代码：

```
#图片路径
img=cv2.imread("pic/1.jpg")
a=[];b=[]
def on_EVENT_LBUTTONDOWN(event, x, y, flags, param):
if event==cv2.EVENT_LBUTTONDOWN:
xy="%d,%d"%(x, y)
a.append(x)
b.append(y)
cv2.circle(img, (x, y), 1, (0, 0, 255), thickness=-1)
cv2.putText(img, xy, (x, y), cv2.FONT_HERSHEY_PLAIN,
1.0, (0, 0, 255), thickness=1)
cv2.imshow("image", img)
cv2.imwrite("pic/result.jpg", img)
cv2.namedWindow("image")
cv2.setMouseCallback("image", on_EVENT_LBUTTONDOWN)
cv2.imshow("image", img)
cv2.waitKey(0)
cv2.destroyWindow()
```



4.打开实验中“**calibration.py**”程序文件，将 **1,2,3** 点的像素坐标和世界坐标填写到程序文件的对应参数中保存即可。

5.后面实验中如果需要重新标定，将更新好参数的“**calibration.py**”和“**Five_Robot_kinematics.py**”拷贝到实验的文件夹中即可。

3D 相机基于深度图像测量人脸距离

一、实验目的

（1）掌握 3D 相机的深度图像采集方法；

（2）掌握 3D 相机的深度图像测距方法；

（3）掌握人脸检测与距离测量方法；

二、实验内容

3D 相机基于深度图像测量人脸距离是通过 RealsenseD415 深度相机采集深度图像检测人脸与摄像头的距离。

三、实验环境

硬件环境	JetsonNX
操作系统	Linux/windows
实验设备	RealsenseD415 深度相机，JetsonNX
实验配件	TypeC

四、实验原理

1.硬件原理

RealsenseD415 的硬件包含了两个深度相机，一个 RGB 相机和一个结构光红外投影仪。深度卷帘相机（逐行扫描），红外结构光深度测距。

2.算法原理

本实验将通过 RGB 相机采集人脸图像后检测人脸得到人脸的检测框，同时也得到了人脸检测框的中心点坐标，将人脸中心点的坐标与深度图像映射就可以得到人脸中心坐标与摄像头的深度值。

3.相关函数

(1)控制云台转动函数：bus_bjdj()函数

bus_bjdj()函数也是封装好的函数，是通过串口发送 16 进制数组方式控制云台舵机的转动，[0xFF,0xFE,0x02,0x01,0x00, 0x01, 0x3C,0x00,0x0D,0x0A]数组从 0 位开始，第五位第六位分别对应舵机的转动的角度值。

五、实验步骤

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）在新建单元格输入“%load 3D 相机人脸检测与测距.py”，点击运行按钮，载入程序。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

参考代码：

```
# 导入库文件
import pyrealsense2 as rs
import numpy as np
import cv2
import time
import serial
import random
```

3.初始化串口

参考代码：

```
data_ser = serial.Serial("/dev/user_robot", 115200, timeout=5)  # 云台串口，设备波特率为 115200
```

4.定义全局变量：人脸检测的中心点

参考代码：

```
global object_x#检测物体中心点 X 的坐标
global object_y#检测物体中心点 Y 的坐标
```

5.云台控制函数

参考代码：

```
#云台控制
def bus_bjdj(value):  # 步进电机位置控制（id，位置）0-315mm
    ddata = [0xFF,0xFE,0x02,0x01,0x00, 0x01, 0x46,0x00,0x0D,0x0A]
    ddata[5] = value
    data_ser.write(ddata)#串口发送数据
```

```
    time.sleep(0.1)
```

6. 获取深度图像和 RGB 图像

参考代码：

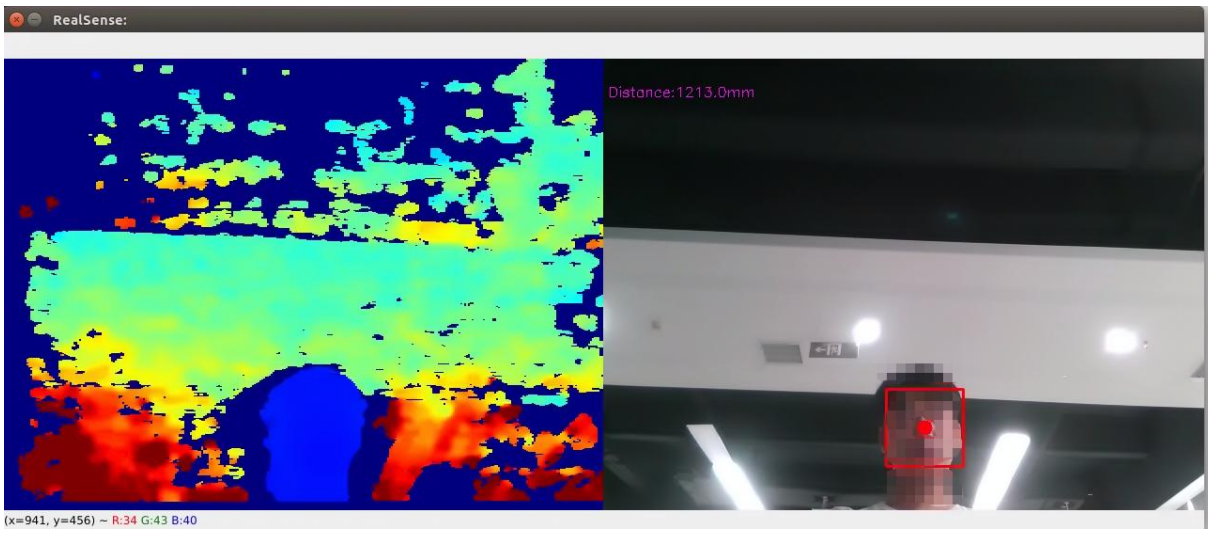
```
if __name__ == '__main__':
    bus_bjdj(0x5A)#控制云台抬起摄像头
    pipeline = rs.pipeline()  # 创建一个管道
    config = rs.config()  # Create a config 并配置要流式传输的管道。
    config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)#使用选定的流参数
    config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
    # Start streaming  开启流
    pipeline.start(config)
    align = rs.align(rs.stream.color) #深度图像向彩色对齐
    print(type(align))
    global object_x
    global object_y
    object_x = 320  # 修改成检测目标的中心点即可
    object_y = 240
    try:
        while True:
            frames = pipeline.wait_for_frames()  # 等待开启通道
            aligned_frames = align.process(frames)  # 将深度框和颜色框对齐
            depth_frame = aligned_frames.get_depth_frame()  # 获得对齐后的帧数深度数据(图)
            color_frame = aligned_frames.get_color_frame()  # 获得对齐后的帧数颜色数据(图)
            img_color = np.asanyarray(color_frame.get_data())  # 把图像像素转化为数组
            img_depth = np.asanyarray(depth_frame.get_data())  # 把图像像素转化为数组
            # Apply colormap on depth image (image must be converted to 8-bit per pixel first)  在深度
图上用颜色渲染
            depth_colormap = cv2.applyColorMap(cv2.convertScaleAbs(img_depth, alpha=0.03),
cv2.COLORMAP_JET)
```

7. 检测人脸中点

参考代码：

```
#人脸检测中点
        image = img_color.copy()
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)#转灰度
        face_detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")#加载人脸检测
模型数据
        faces = face_detector.detectMultiScale(gray, 1.1, 5)#检测人脸
        for x, y, w, h in faces:
            cv2.rectangle(img_color, (x, y), (x + w, y + h), (0, 0, 255), 2)#根据检测的数据画矩形框
            object_x=round(x+w/2)
            object_y=round(y+h/2)
```

```
                print("object_x:",object_x)
                print("object_y:", object_y)
                # 获取目标框内的物体距离，并进行均值滤波
                depth_points = []
                for j in range(50):#取 50 个点的随机数测量平均深度值
                    rand_x = random.randint(x, x + w)
                    rand_y = random.randint(y, y + h)
                    depth_point = round(depth_frame.get_distance(rand_x, rand_y)*1000, 2)
                    if depth_point != 0:
                        depth_points.append(depth_point)
                depth_object = np.mean(depth_points)
                if depth_object >= 300:
                    print("The camera is facing an object mean ", int(depth_object), " mm away.")
                else:
                    print("The camera is facing an object mean <300 mm away.")
                cv2.circle(img_color, (int(object_x), int(object_y)), 8, [0, 0, 255], thickness=-1)#画出中心
点
                cv2.putText(img_color, "Distance:" + str(round(depth_object)) + "mm", (5,
40),cv2.FONT_HERSHEY_SIMPLEX, 0.5, [255, 0, 255])#写出距离值
            image_new = np.hstack((depth_colormap,img_color ))#图像拼接在一起
            cv2.imshow("RealSense:",image_new)
            key = cv2.waitKey(10)
            if key & 0xFF == ord('q') or key == 27:
                cv2.destroyAllWindows()
                break
finally:
    pipeline.stop()#关闭流
```



4

3D 相机人脸检测与云台跟随

一、实验目的

（1）掌握 3D 相机的图像采集方法；

（2）掌握云台控制方法；

（3）掌握人脸检测与云台跟踪方法；

二、实验内容

3D 相机人脸检测与云台跟随是通过 RealsenseD415 深度相机采集图像检测人脸，控制云台相机跟随人脸的移动而移动。

三、实验环境

硬件环境	JetsonNX
操作系统	Linux/windows
实验设备	RealsenseD415 深度相机，Jetson NX
实验配件	TypeC

四、实验原理

1.硬件原理

RealsenseD415 的硬件包含了两个深度相机，一个 RGB 相机和一个结构光红外投影仪。深度卷帘相机（逐行扫描），红外结构光深度测距。

2.算法原理

本实验将通过 RGB 相机采集人脸图像后检测人脸得到人脸的检测框，同时也得到了人脸检测框的中心点坐标，将人脸中心点的坐标与云台转动的角度进行映射就可以得到人脸左右移动时对应的云台舵机应转动的角度，云台就可以跟踪人脸并与人脸保持一致。

3.相关函数

（1）face_detect_demo()

人脸检测函数 face_detect_demo()的用法为：

face_detect_demo()是一个封装好的函数，实现过程是读取图片将图片转为 RGB 格式和灰度图像，调用 opencv 自带的人脸检测模型文件

“haarcascade_frontalface_default.xml”就可以得到人脸的检测框，将人脸检测框画出来即可。

（2）bus_bjdj()

控制云台转动函数 bus_bjdj()也是封装好的函数，是通过串口发送 16 进制数组方式控制云台舵机的转动，[0xFF,0xFE,0x02,0x01,0x00, 0x01, 0x3C,0x00,0x0D,0x0A]数组从 0 位开始，第五位第六位分别对应舵机的转动的角度值。

五、实验步骤

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）在新建单元格输入“%load 3D 相机人脸检测与云台跟随.py”，点击运行按钮，载入程序。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

相机采集程序 UI 基于 pyQt5 进行编写。

参考代码：

```
# 导入库文件
import sys
from mainui import Ui_MainWindow
import cv2
import time
from PyQt5.QtWidgets import *
from PyQt5 import QtCore, QtGui, QtWidgets
import serial
```

3.初始化串口

参考代码：

```
data_ser = serial.Serial("/dev/user_robot", 115200, timeout=5)  # 云台接收的串口，设备波特率为115200
```

4.界面类初始化 ：并将 UI 控件与事件函数进行绑定

参考代码：

```
global value,x
value = 90
class Mywindow(QMainWindow, Ui_MainWindow):
    def __init__(self):  # 创建构造函数
        super().__init__()  # 调用父类函数，继承
        self.timer_camera = QtCore.QTimer()  # 定时器
        # 数据帧
        self.setupUi(self)  # 调用 UI
        self.cap = cv2.VideoCapture(2)  # 准备获取图像
        ret,image=self.cap.read()
        if ret!=True:
            self.CAM_NUM = 3
        else:
            self.CAM_NUM = 2
        self.slot_init()  # 设置槽函数

    def slot_init(self):
        # 设置槽函数
        self.pushButton_open.clicked.connect(self.button_open_camera_click)
        self.timer_camera.timeout.connect(self.face_detect_demo)
        self.pushButton_close.clicked.connect(self.closeEvent)
```

5.事件函数:开启采集图像定时器

参考代码：*

```
# 开启采集定时器
def button_open_camera_click(self):
    self.bus_bjdj(0x5A)  # 使云台抬升
    if self.timer_camera.isActive() == False:
        flag = self.cap.open(self.CAM_NUM)
        if flag == False:
            msg = QtWidgets.QMessageBox.warning(
                self, u"Warning", u"请检测相机与电脑是否连接正确",
                buttons=QtWidgets.QMessageBox.Ok,
                defaultButton=QtWidgets.QMessageBox.Ok)
        else:
            self.timer_camera.start(10)
```

6. 关闭相机函数

参考代码：*

```
# 关闭相机显示
def closeEvent(self):
    if self.timer_camera.isActive() != False:
        ok = QtWidgets.QPushButton()
```

```
            cacel = QtWidgets.QPushButton()
            msg = QtWidgets.QMessageBox(QtWidgets.QMessageBox.Warning, u"关闭", u"是否关闭！ ")
            msg.addButton(ok, QtWidgets.QMessageBox.ActionRole)
            msg.addButton(cacel, QtWidgets.QMessageBox.RejectRole)
            ok.setText(u'确定')
            cacel.setText(u'取消')
            if msg.exec_() != QtWidgets.QMessageBox.RejectRole:
                if self.cap.isOpened():
                    self.cap.release()
                if self.timer_camera.isActive():
                    self.timer_camera.stop()
                self.label_imput_image.setText(
                    "<html><head/><body><p align=\"center\"><img
src=\":/newPrefix/pic/Hint.png\"/><span style=\" font-size:28pt;\">点击打开云台跟踪
</span><br/></p></body></html>")
```

7. 人脸检测函数

参考代码：

```
 # 人脸检测
def face_detect_demo(self):
    flag, self.image = self.cap.read()
    self.image = cv2.flip(self.image, 1)  # 左右翻转
    imageshow = self.image.copy()
    imageshow = cv2.cvtColor(imageshow, cv2.COLOR_BGR2RGB)
    gray = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
    face_detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
    faces = face_detector.detectMultiScale(gray, 1.1, 5)
    global value,x
    for x, y, w, h in faces:
        cv2.rectangle(imageshow, (x, y), (x + w, y + h), (0, 0, 255), 2)
        print("x:",x+w/2)
        if x+w/2 >= 0 and x+w/2 <= 270:
            value = value - 1
            if value <= 20:
                value = 20
        if x+w/2 >= 380 and x+w/2 <= 640:
            value = value + 1
            if value >= 160:
                value = 160
        print("value:",value)
    self.bus_bjdj(int(value))
    recImage = QtGui.QImage(imageshow.data, imageshow.shape[1], imageshow.shape[0],
QtGui.QImage.Format_RGB888)
```

```
        self.label_imput_image.setPixmap(QtGui.QPixmap.fromImage(recImage))
        self.label_imput_image.setScaledContents(True)
```

8. 云台控制函数

参考代码：

```
# 控制云台转动
def bus_bjdj(self, value):   # 步进电机位置控制（id，位置）0-315mm
        ddata = [0xFF, 0xFE, 0x02, 0x01, 0x00, 0x01, 0x46, 0x00, 0x0D, 0x0A]
        ddata[5] = value
        data_ser.write(ddata)   # 串口发送数据
        time.sleep(0.1)
```

9. 界面启动函数

参考代码：

```
if __name__ == "__main__":
        app = QApplication(sys.argv)
        mywindow = Mywindow()
        mywindow.show()
        app.exec_()
```

人脸录入与识别

一、实验目的

（1）掌握人脸图像采集方法；

（2）掌握人脸图像训练方法；

（3）掌握人脸识别方法；

二、实验内容

人脸录入与识别是通过 RealsenseD415 RGB 相机采集图像后检测人脸，将人脸数据进行训练，然后识别人脸。

三、实验环境

硬件环境	JetsonNX
操作系统	Linux/windows
实验设备	RealsenseD415 深度相机，Jetson NX
实验配件	TypeC

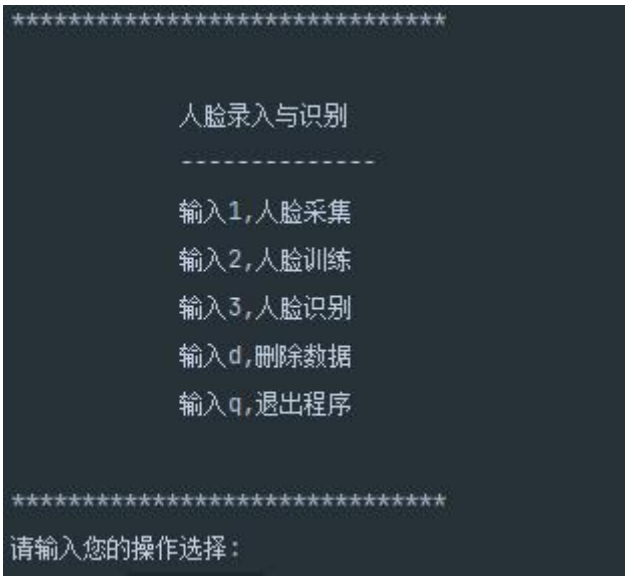
四、实验原理

1.硬件原理

RealsenseD415 的 RGB 相机采集人脸彩色图像，采集人脸图像后进行人脸检测，将检测人脸照片(程序中默认设置 10 张，数量可以根据需求进行添加)保存在 dataset 文件夹。

2.算法原理

本实验将通过 RGB 相机采集人脸图像通过 opencv 自带的人脸检测模型检测人脸得到人脸的图片并保存 dataset 文件夹，默认保存 10 张照片，保存照片后，选择对人脸图片进行训练，将训练好的人脸数据保存到 trainer 文件夹下的 trainer.yml 文件中，人脸识别就通过调用训练的 trainer.yml 模型进行人脸识别，Main.py 程序文件是主程序的入口负责程序流程和功能选择，人脸采集程序是 datasets.py，人脸训练程序是 training.py，人脸识别是 recognition.py，数据清除是 delFile.py，Main.py 主程序通过不同的选择进行不同的操作。如下所示：



注意：输入的姓名请使用英文字母缩写或者拼音，程序中没有做中文字体适配，如果输入中文字体程序会报错。

3.相关函数

（1）datasets()

检测人脸并采集人脸图像 datasets()函数是封装好的函数，是调用 opencv 自带的人脸模型“data/haarcascade_frontalface_default.xml”进行人脸检测，将检测到的人脸图片保存到 dataset 文件夹中，默认采集 10 张人脸数据，如果需要提高准确率可以将人脸的采集数据进行增加。

（2）training()

人脸数据训练函数 Training()是利用 opencv 自带的 LBPH 方法进行人脸识别，LBPH（Local Binary Patterns Histogram，局部二值模式直方图）所使用的模型基于 LBP(局部二值模式)算法。LBP 最早是被作为一种有效的纹理描述算子提出的，由于在表述图像局部纹理特征上效果明显，而得到广泛应用。

LBP 算法的基本原理是，将像素点 A 的值与其最邻近的 8 个像素点的值逐一比较：

如果 A 的像素值大于其临近点的像素值，则得到 0

如果 A 的像素值小于其临近点的像素值，则得到 1

最后，将像素点 A 与其周围 8 个像素点比较所得到的 0、1 值连接起来，得到一个 8 位的二进制序列，将该二进制序列转换为十进制数作为点 A 的 LBP 值。

在 OpenCV 中，提供函数 cv2.face.LBPHFaceRecognizer_create()来生成 LBPH 识别器实例模型，利用 train()完成训练。

def LBPHFaceRecognizer_create(radius=None, neighbors=None, grid_x=None, grid_y=None, threshold=None):

radius：半径值，默认 1

neighbors：领域点的个数，默认采用 8 领域，根据需要可以计算更多的领域点

grid_x：将 LBP 特征图像划分为一个个单元格时，每个单元格在水平方向上的像素个数。默认值 8，即将 LBP 特征图像在行方向上以 8 个像素为单位分组

grid_y：将 LBP 特征图像划分为一个个单元格时，每个单元格在垂直方向上的像素个数。默认值 8，即将 LBP 特征图像在列方向上以 8 个像素为单位分组

threshold：预测时所使用的阈值。如果大于该阈值，就认为没有识别到任何目标对象

cv2.face_FaceRecognizer.train(self, src, labels)

src：训练图像，相当于前面识别图像的训练集，用来学习的人脸图像

labels：标签，人脸图像所对应的标签

（3）recognition()

人脸识别函数 recognition()是利用 predict()函数完成人脸识别。

cv2.face_FaceRecognizer.predict(self, src)

src：需要识别的人脸图像

Predict()返回值有两个，一个返回前面训练集匹配的人脸识别的标签 label，另一个是用来衡量识别结果与原有模型之间的距离。通常情况下，小于 80 的值是可以接受的，如果该值大于 80 则认为差别较大。

五、实验步骤

Main 主程序

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）在新建单元格输入“%load Main.py”，点击运行按钮，载入程序。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

参考代码：

```
# 导入库文件
from recognition import recognition
```

```
from training import training
from datasets import datasets
from delFile import del_file
import serial
import time
```

3.初始化串口

参考代码：

```
data_ser = serial.Serial("/dev/user_robot", 115200, timeout=5)  # 插入传感器自动识别设备，并设置波特率
```

4.人脸检测与识别流程控制：选择人脸采集，人脸训练，人脸识别等

参考代码：

```
def main():
    bus_bjdj(0x5A)
    facedict = {}
    cur_path = r'./dataset/'
    while True:
        print('*' * 31)
        print('''
            人脸录入与识别
            --------------
            输入 1,人脸采集
            输入 2,人脸训练
            输入 3,人脸识别
            输入 d,删除数据
            输入 q,退出程序
        ''')
        print('*' * 31)
        num = input("请输入您的操作选择: ")
        if num == '1':
            mydict = datasets()
            facedict.update(mydict)
            print(facedict)
        elif num == '2':
            training()
        elif num == '3':
            recognition(facedict)
        elif num == 'd':
            del_file(cur_path)
        elif num == 'q':
            print("退出程序成功!")
            break
        else:
```

```
            print("您输入有误,请重新输入!")

def bus_bjdj(value):   # 步进电机位置控制（id，位置）0-315mm
    ddata = [0xFF,0xFE,0x02,0x01,0x00,0x01,0x46,0x00,0x0D,0x0A]
    ddata[5] = value
    data_ser.write(ddata)
    time.sleep(0.1)
if __name__ == '__main__':
    main()
```

datasets 人脸采集程序

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）在新建单元格输入“%load datasets.py”，点击运行按钮，载入程序。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

参考代码：

```
# 导入库文件
import cv2
```

3.人脸数据采集：检测人脸，保存人脸图片

参考代码：

```
# 人脸数据采集
def datasets():
    print('准备开始人脸数据采集')
    mydict = {}
    while True:
        print("输入'q'停止添加")
        face_id = input('请设置新的人脸 id(id 为数字)：')
        if face_id == 'q':
            break
        face_name = input('请输入新的人脸 name(name 为英文或字母)：')
        if face_name == 'q':
            break
```

```
            mydict[face_id] = face_name
            # print(mydict)
            count = 0
            cam = cv2.VideoCapture(2)   # 打开相机
            ret, image = cam.read()
            if ret != True:
                cam = cv2.VideoCapture(3)
            face_detector = cv2.CascadeClassifier('data/haarcascade_frontalface_default.xml')#加载人脸检
测模型

            while True:
                _, image_frame = cam.read()
                gray = cv2.cvtColor(image_frame, cv2.COLOR_BGR2GRAY)
                faces = face_detector.detectMultiScale(gray, 1.3, 5)#检测人脸
                for (x, y, w, h) in faces:
                    cv2.rectangle(image_frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
                    count += 1
                    cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg", gray[y:y + h, x:x +
w])
                cv2.imshow('frame', image_frame)
                if cv2.waitKey(100) & 0xFF == ord('q'):
                    break
                elif count > 10:
                    print('%s:人脸数据采集完成！' % (mydict[face_id]))
                    break
            cam.release()
            cv2.destroyAllWindows()
        print(mydict)
        return mydict
```

training 人脸训练程序

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）在新建单元格输入“%load training.py”，点击运行按钮，载入程序。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

参考代码：

```
# 导入库文件
import os
import cv2
import numpy as np
from PIL import Image
```

3.训练人脸数据：获取人脸图像和标签，训练人脸，保存数据

参考代码：

```
path = r'./trainer'#保存训练数据的路径
# 人脸数据训练
def training():
    recognizer = cv2.face.LBPHFaceRecognizer_create()#LBP 识别器
    detector = cv2.CascadeClassifier("data/haarcascade_frontalface_default.xml")
    #获取图片和标签
    def getImagesAndLabels(path):
        imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
        faceSamples = []
        ids = []
        for imagePath in imagePaths:
            PIL_img = Image.open(imagePath).convert('L')
            img_numpy = np.array(PIL_img, 'uint8')
            id = int(os.path.split(imagePath)[-1].split(".")[1])
            # print(id)
            faces = detector.detectMultiScale(img_numpy)
            for (x, y, w, h) in faces:
                faceSamples.append(img_numpy[y:y + h, x:x + w])
                ids.append(id)
        return faceSamples, ids

    faces, ids = getImagesAndLabels('./dataset')
    recognizer.train(faces, np.array(ids))#训练人脸
    recognizer.save('trainer/trainer.yml')#保存训练好的人脸数据
    print("数据训练完毕！ ")
```

recognition 人脸识别程序

1.运行 jupyter lab

（1）打开桌面的“实验”文件夹，在空白处单击鼠标右键，再点击“在终端打开”，在终端界面输入“jupyter lab”；

（2）在 jupyter lab 编程界面选择 Notebook 下 Python3，进入程序编辑器；

（3）鼠标右键单击“未命名.ipynp”的新建程序块，选择“Rename”将程序名命为实验名称。

（4）在新建单元格输入“%load recognition.py”，点击运行按钮，载入程序。

2.导入库文件及函数

导入 cv2 相关库，利用 cv2 的相关库中的对象与函数引用图像处理和计算机视觉方面的各种通用算法。

参考代码：

```
# 导入库文件
import cv2
```

3.识别人脸数据：加载训练的人脸模型数据，检测人脸，识别人脸

参考代码：

```
# 人脸识别
def recognition(mydict):
    print("按'q'退出！ ")
    mydict = mydict
    recognizer = cv2.face.LBPHFaceRecognizer_create()#LBP 识别器
    recognizer.read('trainer/trainer.yml')#读取训练的人脸数据
    cascadePath = "data/haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(cascadePath)#加载自带的人脸检测模型
    font = cv2.FONT_HERSHEY_SIMPLEX
    cam = cv2.VideoCapture(2)#打开相机
    ret, image = cam.read()
    if ret != True:
        cam = cv2.VideoCapture(3)
    while True:
        ret, im = cam.read()
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(gray, 1.2, 5)#检测人脸
        for (x, y, w, h) in faces:
            cv2.rectangle(im, (x - 20, y - 20), (x + w + 20, y + h + 20), (0, 255, 0), 4)
            Id, conf = recognizer.predict(gray[y:y + h, x:x + w])#识别人脸
            if conf < 80:#大于 80 则认为差别大
```

```
                    if str(Id) in mydict:
                        Id = mydict[str(Id)]
                else:
                    Id = "Unknow"
                cv2.rectangle(im, (x - 22, y - 90), (x + w + 22, y - 22), (0, 255, 0), -1)
                cv2.putText(im, str(Id), (x, y - 40), font, 2, (255, 255, 255), 3)
        cv2.imshow('im', im)
        if cv2.waitKey(100) & 0xFF == ord('q'):
            break
cam.release()
cv2.destroyAllWindows()
```