




# 1.实验的提交：已提交到 Git 仓库：


[zhaojiacheng999/HFUT-CV-2021214585 \(github.com\)](https://github.com/zhaojiacheng999/HFUT-CV-2021214585)


```
28082@DESKTOP-BIGMKV MINGW64 ~/Desktop/计算机视觉 (master)
$ git push CV master
Enumerating objects: 35, done.
Counting objects: 100% (35/35), done.
Delta compression using up to 8 threads
Compressing objects: 100% (31/31), done.
Writing objects: 100% (35/35), 47.34 MiB | 996.00 KiB/s, done.
Total 35 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/zhaojiacheng999/HFUT-CV-2021214585/pull/new/master
remote:
remote: To https://github.com/zhaojiacheng999/HFUT-CV-2021214585.git
* [new branch]      master -> master
```


 **HFUT-CV-2021214585** Public Pin Unwatch


 main
















 2 Branches

 0 Tags

 Add file

 Code

 **zhaojiacheng999** CV 1df40fb · 36 minutes ago 1 Commits

	.idea	CV	36 minutes ago
	data/mnist/MNIST/raw	CV	36 minutes ago
	2023机器视觉课程实验及实验报告要求_20231...	CV	36 minutes ago
	assigment1_test.png	CV	36 minutes ago
	assigment2_out.mp4	CV	36 minutes ago
	assigment2_outTEMP_MPY_wvf_snd.mp3	CV	36 minutes ago
	assigment2_test.jpg	CV	36 minutes ago
	assigment2_test.mp4	CV	36 minutes ago
	assigment3_test.png	CV	36 minutes ago
	assigment_1.py	CV	36 minutes ago
	assigment_2.py	CV	36 minutes ago
	assigment_3.py	CV	36 minutes ago
	model_mnist.pth	CV	36 minutes ago
	test.jpg	CV	36 minutes ago
	texture_features.npy	CV	36 minutes ago

# 2.环境的命名与设置

使用的环境为以名字缩写命名的 Python3.9 环境

```
Python 解释器: Python 3.9 (zjc) C:\Users\28082\anaconda3\envs\zjc\python.exe
```

## 课程实验一：图像滤波

### 一、实验目的：

使用 Sobel 算子、给定卷积核滤波自己拍摄的图像，并提取图像的颜色直方图和纹理特征其中，给定卷积核：

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

具体要求：

- 任务输入：自己拍摄的图像。
- 任务输出：经过 Sobel 算子滤波的图像，经过给定卷积核滤波的图像，可视化图像的颜色直方图，保存纹理特征至 npy 格式。
- 滤波、直方图计算、纹理特征提取过程不可以调用函数包。
- 代码语言不限，纹理特征提取方法不限，要求提交整个算法源代码，实验结果（算法输入图片、输出图片、直方图，纹理结果），算法分析等内容。

### 二、实验原理

#### 2.1 Sobel 算子边缘监测

索伯算子（Sobel）是图像处理中的算子之一，有时又称为索伯-费尔德曼算子或索伯滤波器，在影像处理及电脑视觉领域中常被用来做边缘检测。索伯算子最早是由美国计算机科学家艾尔文·索伯及盖瑞·费尔德曼（Gary Feldman）于 1968 年在史丹佛大学的人工智能实验室（SAIL）所提出，因此为了表扬他们的贡献，而用他们的名字命名。在技术上，它是一离散性差分算子，用来运算图像亮度函数的梯度之近似值。在图像的任何一点使用此算子，索伯算子的运算将会产生对应的梯度向量或是其范数。概念上，索伯算子就是一个小且是整数的滤波器对整张影像在水平及垂直方向上做卷积，因此它所需的运算资源相对较少，另一方面，对于影像中的频率变化较高的地方，它所得的梯度之近似值也比较粗糙。

该算子包含两组  $3 \times 3$  的矩阵，分别为横向及纵向，将之与图像作平面卷积，即按所示方程式计算：

$$\left( \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \right) [2,2] = \\ (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + \\ (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + \\ (c \cdot 7) + (b \cdot 8) + (a \cdot 9)$$

即可分别得出横向及纵向的亮度差分近似值。如果以 $\mathbf{A}$ 代表原始图像， $\mathbf{G}_x$ 及 $\mathbf{G}_y$ 分别代表经横向及纵向边缘检测的图像，其公式如下：

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

图像的每一个像素的横向及纵向梯度近似值可用以下的公式结合，来计算梯度的大小。

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$

然后可用以下公式计算梯度方向。

$$\Theta = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

## 2.2 颜色直方图

颜色直方图用以反映图像颜色的组成分布，即各种颜色出现的概率。Swain 和 Ballard 最先提出了应用颜色直方图进行图像特征提取的方法[40]，首先利用颜色空间三个分量的剥离得到颜色直方图，之后通过观察实验数据发现将图像进行旋转变换、缩放变换、模糊变换后图像的颜色直方图改变不大，即图像直方图对图像的物理变换是不敏感的。因此常提取颜色特征并用颜色直方图应用于衡量和比较两幅图像的全局差。另外，如果图像可以分为多个区域，并且前景与背景颜色分布具有明显差异，则颜色直方图呈现双峰形。

颜色直方图也有其缺点：由于颜色直方图是全局颜色统计的结果，因此丢失了像素点间的位置特征。可能有几幅图像具有相同或相近的颜色直方图，但其图像像素位置分布完全不同。因此，图像与颜色直方图得多对一关系使得颜色直方图在识别前景物体上不能获得很好的效果。

## 2.3 纹理特征提取--GLCM

灰度共生矩阵(GLCM)的统计方法是 20 世纪 70 年代初由 R.Haralick 等人提出的，它是在假定图像中各像素间的空间分布关系包含了图像纹理信息的前提下，

提出的具有广泛性的纹理分析方法。

图像  $I(N \times N)$  的任意一点  $f(x,y)$  与其偏离点  $f(x+a,y+b)$ ，设该点对的灰度值为  $(g1,g2)$ 。令点  $f(x,y)$  在整个图像上移动，便得到所有  $(g1,g2)$  的值。假设灰度值的级数为  $k$ ，则  $(g1,g2)$  共有  $k^2$  种组合。对整个画面，统计每种  $(g1,g2)$  值出现的次数，排列为方阵并将其归一化，这个方阵称为灰度共生矩阵。

差分值的选取：距离差分值  $(a,b)$  取不同的数值组合，可得到不同情况下的联合概率矩阵。通常  $(a,b)$  是按照纹理周期分布的特性来选择的，对于较细的纹理，选取  $1,0,0,1,(2,0)$  等较小的差分值。

### 三、实验方法

#### 1. Sobel 滤波器实现

思路：**Sobel** 算子是一种用于边缘检测的算法。它通过计算图像亮度的空间梯度来突出显示边缘。在本实验中，我们将实现水平和垂直方向的 **Sobel** 滤波器，然后将这两个方向的结果合成来得到最终的边缘检测结果。

实现：对于每个像素，我们将使用 **Sobel** 算子与其周围的像素值进行卷积。这意味着我们将 **Sobel** 算子的每个元素乘以对应像素周围的值，然后将它们相加。这样我们就能得到该像素在特定方向上的梯度值。

#### 2. 给定卷积核的滤波实现

思路：给定的卷积核是一个简单的  $3 \times 3$  矩阵，用于在图像上执行自定义滤波。该过程类似于 **Sobel** 滤波，但使用不同的核。

实现：我们将使用给定的卷积核，通过与图像中每个像素及其邻居的卷积来修改图像。这将改变图像的某些特性，可能会突出或抑制某些特征。

#### 3. 颜色直方图的计算和可视化

思路：颜色直方图展示了图像中每种颜色的频率。这对于分析图像的颜色分布非常有用。

实现：我们将计算图像中每个颜色值的出现次数，并将这些数据可视化为直方图。这需要遍历图像中的所有像素，并记录每种颜色的出现次数。

#### 4. 纹理特征提取

思路：纹理特征描述了图像的纹理信息，它们可以从图像的局部区域中提取出来。

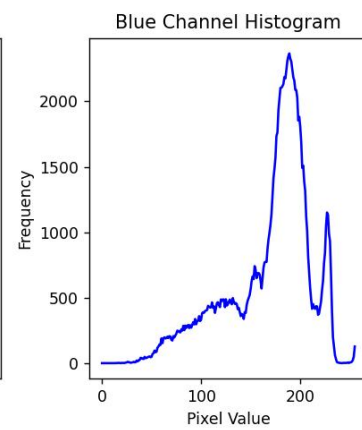
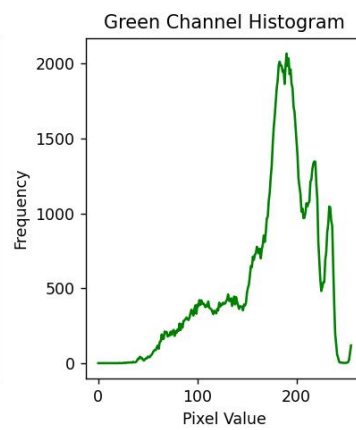
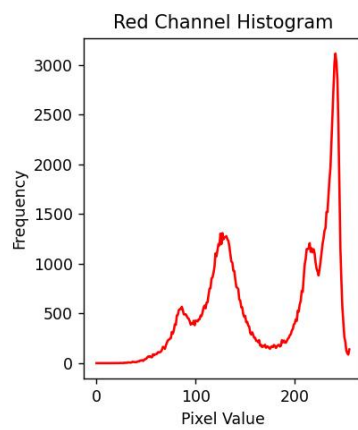
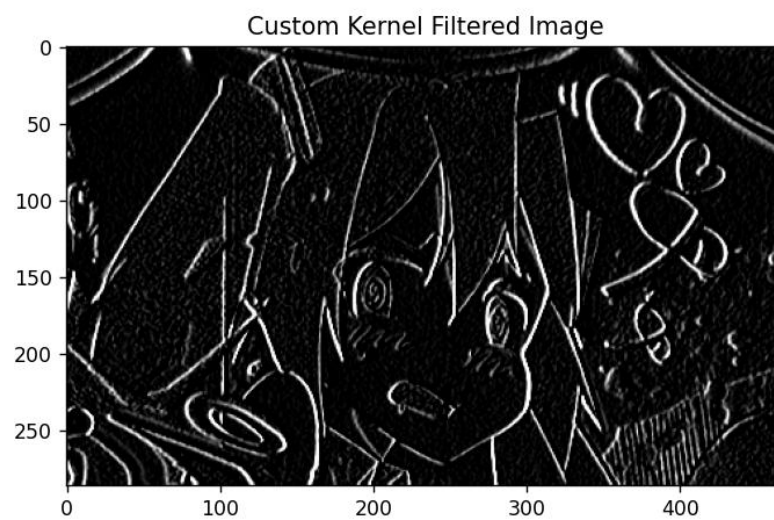
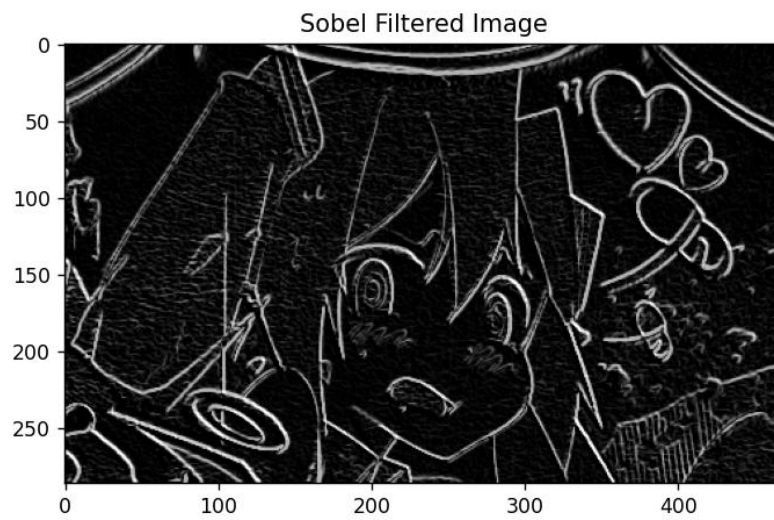
实现：这个步骤的具体实现取决于选择哪种纹理特征提取方法。一种常见的方法是使用灰度共生矩阵（**GLCM**）来分析纹理模式。我们将计算一些统计量，如对比度、同质性等，来描述图像的纹理。

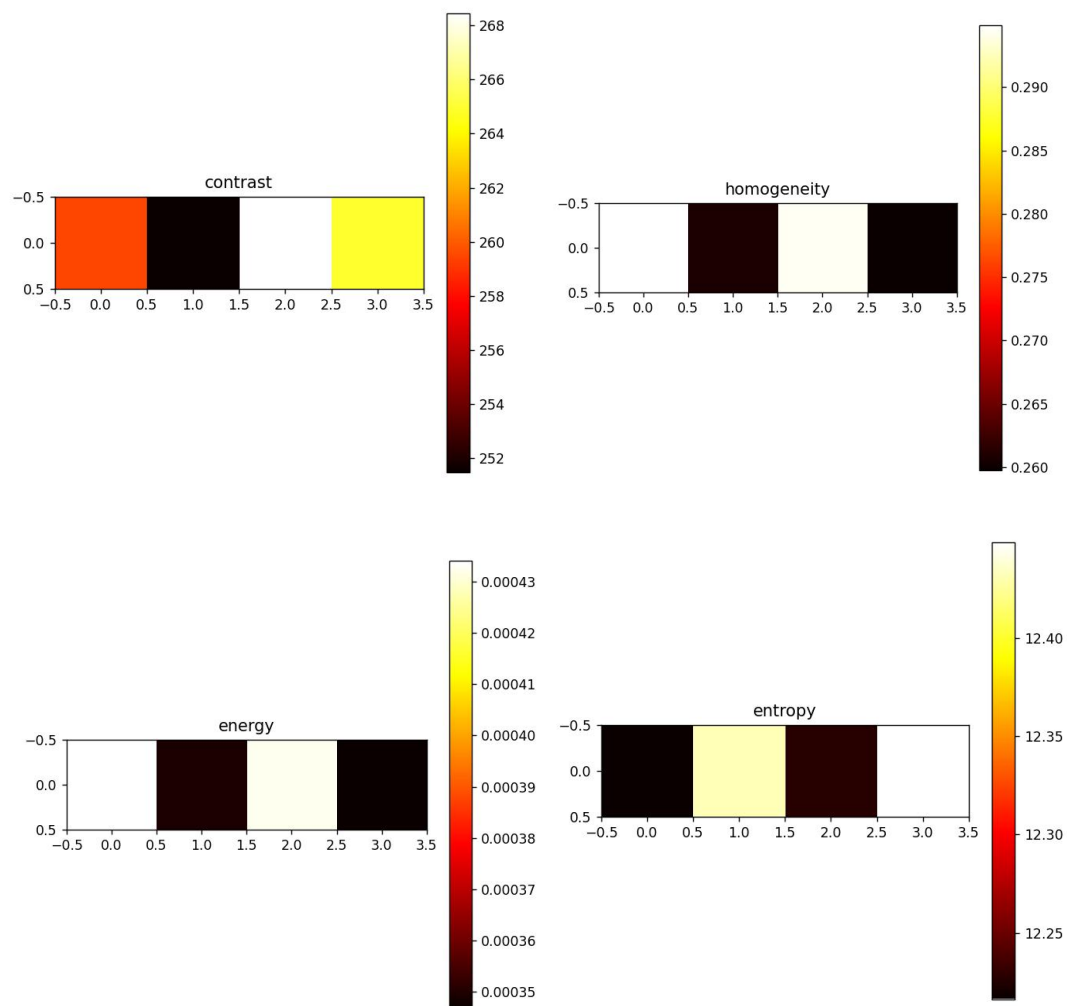
#### 5. 保存纹理特征

实现：一旦计算出纹理特征，我们可以将它们保存为 **numpy** 格式的文件，这是

一种用于存储数组数据的格式。

#### 四、实验结果





```
contrast:
[[259.49339549 251.47892118 268.45201068 264.88596478]]
homogeneity:
[[0.29487672 0.2606562 0.29434464 0.25977856]]
energy:
[[0.00043407 0.00034914 0.00043245 0.0003468 ]]
entropy:
[[12.21623792 12.43212743 12.22624377 12.44849742]]
```

## 五、实验体会

### 实验设置

编程语言：选择 Python 进行实验，因其广泛应用于图像处理和其丰富的库支持。

开发环境：使用 Pycharm，它提供了一个交互式的开发环境，便于代码编写和结果展示。



硬件和软件环境：实验在配备中等规格 CPU 的个人计算机上进行，操作系统为 Windows 11，Python 版本为 3.8。

### 实验过程与结果分析

**Sobel 滤波器的实现：**实现了 Sobel 算子来检测图像边缘，这个过程加深了我对图像梯度和边缘检测概念的理解。

**自定义卷积核的应用：**使用给定的卷积核进行图像滤波。这一步让我理解了如何手动应用卷积核于图像处理。

**颜色直方图的提取与可视化：**计算并可视化了图像的颜色直方图，这帮助我更好地理解图像的颜色分布。

**纹理特征的提取：**提取了图像的纹理特征，这一步是挑战性的，但也非常有教育意义。

**Sobel 滤波器和自定义卷积核：**两种方法都成功地描绘了图像的特性，但 Sobel 算子在突出边缘方面表现更佳。**颜色直方图：**直方图清晰地展示了图像中颜色的分布，这在视觉上很直观。**纹理特征：**提取的纹理特征有助于进一步分析图像的细节和质地。

在这次实验中，我深入探索了图像处理的基础概念，包括使用 Sobel 算子和自定义卷积核进行图像滤波，以及计算和可视化颜色直方图和提取纹理特征。整个过程是在 Python 环境下进行的，没有使用任何高级图像处理库，这对我理解和掌握图像处理的基本原理非常有帮助。通过手动实现这些算法，我不仅加深了对图像边缘检测、颜色分布和纹理分析的理解，也提高了我的编程和问题解决能力。尽管过程中遇到了一些挑战，比如调整算法参数以优化结果，但这些挑战最终都转化为了宝贵的学习经验。我认识到虽然手动实现图像处理算法很有教育意义，但在实际应用中，使用成熟的图像处理库可能更加高效。这次实验不仅增强了我对图像处理技术的理解，也激发了我对未来学习和应用这些技术的兴趣。

## 课程实验二：车道线检测

### 一、实验目的：

车道线检测是自动驾驶的基本模块。请使用霍夫变换实现车道线的检测。

具体要求：

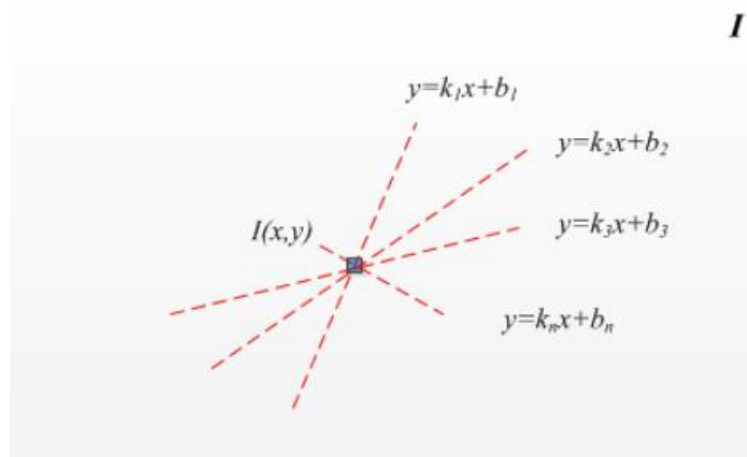
- 任务输入：自己拍摄的校园中道路图像（画有车道线的路）。
- 任务输出：图像中车道线的位置（如右图）。
- 代码语言不限，方法不限，要求提交整个算法源代码，模型结果，算法分析等内容。

## 二、实验原理

### 2.1 霍夫变换的基本思想

在计算机中，图像一般被表示为二维矩阵  $I$ 。图像中像素  $I(x, y)$  所反映的位置信息即图像像素的空间信息，而像素值则是该空间位置下亮度信息的反映。霍夫变换所针对的是图像中形状特征的检测，所用到的是图像的像素排列的空间信息。所以，霍夫变换输入的图像要求为二值图像（仅反映图像中待检测的感兴趣像素集），一般为阈值变换或边缘检测算子处理后的输出。

下面以霍夫直线检测为例谈谈霍夫变换的基本思想。如图 1 所示，在图像  $I$  中通过像素  $I(x, y)$  的直线共有  $n$  条，那么将这些直线所对应的直线参数映射到由这些直线参数张成的二维参数空间  $\{K, B\}$  中，可以形成一条曲线。而若将存在直线的二值图像中所有像素点都用类似方法将其对应直线映射到二维参数空间中去，并对参数频次进行统计叠加（类似灰度直方图，但这里参数不再是一维灰度，而是二维直线参数），那么可以形成一簇明暗不均的曲线：曲线上的像素点越明亮，说明该像素对应参数所生成的直线在原图像中通过的感兴趣像素点越多，则该直线为所期望探测的直线概率越大。不难看出，霍夫变换实际上是一种统计投票决策算法，是利用参数空间域中参数“拟合”直线的过程。



根据上述思想，我们不难得到霍夫变换算法的基本流程：

1. 初始化参数空间，将参数空间矩阵每一个像素置 0；
2. 遍历图像当中不为 0 的像素点，记录所有经过该像素点直线的直线参数，



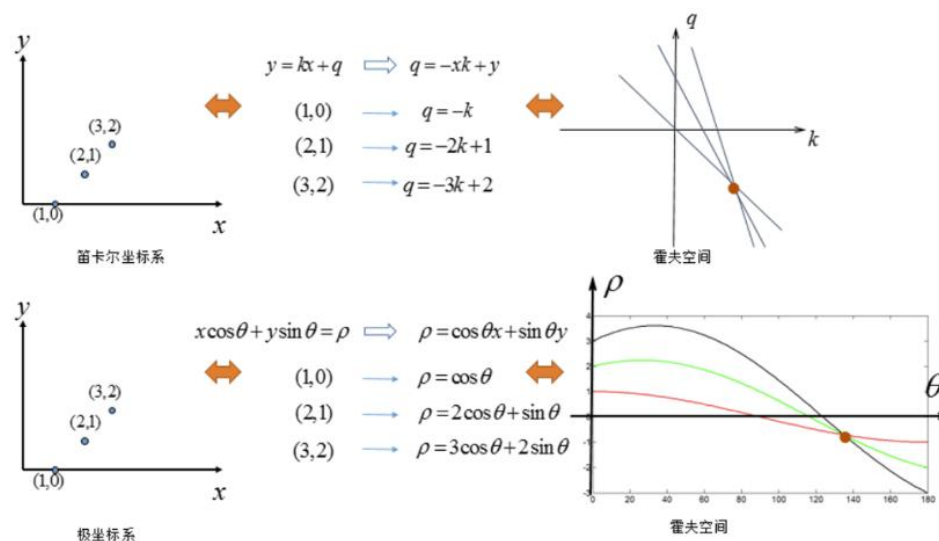
将参数空间中对应该直线参数的像素值加 1;

3.根据先验知识,分析参数矩阵,获取参数矩阵某些峰值作为检测结果。

注意到,直线方程所构成的参数空间对竖直方向的直线难以表达(斜率无穷大),且斜率分量难以均匀量化。故在实际应用中,我们一般采用先将直线方程转换为直线极坐标方程,再利用极坐标方程建立参数矩阵的方式避免上述问题。即先将坐标点 $(x,y)$ 转化为 $(\rho,\theta)$ ,再映射到 $\{R,\Theta\}$ 参数空间。直线的参数方程为

$$\rho = x \cos \theta + y \sin \theta$$

经过变换,图像空间中的每个点 $(x,y)$ 就被映射为一个 $(r,\theta)$ 极坐标空间中的正弦曲线。而图像空间直角坐标系中共线的点所对应的参数空间中正弦曲线相交于一点 $(r',\theta')$ 。这样就把在图像空间中检测直线的问题,转化为在极坐标参数空间中寻找通过点 $(r,\theta)$ 的最多正弦曲线数量的问题。霍夫空间中,曲线的交点次数越多,所代表的参数越确定(相交的曲线都是图像空间直角坐标系上的点),画出的图形越饱满。空间直角坐标系上一条直线上的所有点都转化为参数空间上的曲线,曲线一定会交于一个共同点



## 2.2 Canny 边缘检测

在进行实验之前,我们需要先对图像进行边缘检测。Canny 边缘检测算法可以分为以下 5 个步骤:

### 1. 应用高斯滤波来平滑(模糊)图像,目的是去除噪声

高斯滤波器是将高斯函数离散化,将滤波器中对应的横纵坐标索引代入到高斯函数,从而得到对应的值。

二维的高斯函数如下：其中  $(x, y)$  为坐标， $\sigma$  为标准差

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

不同尺寸的滤波器，得到的值也不同，下面是  $(2k+1) \times (2k+1)$  滤波器的计算公式：

$$H[i, j] = \frac{1}{2\pi\sigma^2} e^{-\frac{(i-k-1)^2+(j-k-1)^2}{2\sigma^2}}$$

常见的高斯滤波器大小为  $5 \times 5$ ， $\sigma = 1.4$ ，其近似值为：

$$K = \frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

## 2. 计算梯度强度和方向

接下来，我们要寻找边缘，即灰度强度变化最强的位置，（一道黑边一道白边中间就是边缘，它的灰度值变化是最大的）。在图像中，用梯度来表示灰度值的变化程度和方向。

常见方法采用 Sobel 滤波器【水平  $x$  和垂直  $y$  方向】在计算梯度和方向。

$$G = \sqrt{(G_x^2 + G_y^2)}$$

$$\theta = \arctan \frac{G_y}{G_x}$$

## 3. 应用非最大抑制技术 NMS 来消除边误检

原理：遍历梯度矩阵上的所有点，并保留边缘方向上具有极大值的像素

这一步的目的是将模糊（blurred）的边界变得清晰（sharp）。通俗的讲，就是保留了每个像素点上梯度强度的极大值，而删掉其他的值。对于每个像素点，进行如下操作：

a) 将其梯度方向近似为以下值中的一个（0,45,90,135,180,225,270,315）（即上下左右和 45 度方向）

- b) 比较该像素点，和其梯度方向正负方向的像素点的梯度强度
- c) 如果该像素点梯度强度最大则保留，否则抑制（删除，即置为 0）

$$M_T(m, n) = \begin{cases} M(m, n), & \text{if } M(m, n) > T \\ 0, & \text{otherwise} \end{cases}$$

#### 4. 应用双阈值的方法来决定可能的（潜在的）边界

这个阶段决定哪些边缘是真正的边缘，哪些边缘不是真正的边缘。

经过非极大抑制后图像中仍然有很多噪声点。Canny 算法中应用了一种叫双阈值的技术。即设定一个阈值上界 maxVal 和阈值下界 minVal，图像中的像素点如果大于阈值上界则认为必然是边界（称为强边界，strong edge），小于阈值下界则认为必然不是边界，两者之间的则认为是候选项（称为弱边界，weak edge），需进行进一步处理——如果与确定为边缘的像素点邻接，则判定为边缘；否则为非边缘。

#### 5. 利用滞后技术来跟踪边界

这个阶段是进一步处理弱边界。

大体思想是，和强边界相连的弱边界认为是边界，其他的弱边界则被抑制。

由真实边缘引起的弱边缘像素将连接到强边缘像素，而噪声响应未连接。为了跟踪边缘连接，通过查看弱边缘像素及其 8 个邻域像素，只要其中一个为强边缘像素，则该弱边缘点就可以保留为真实的边缘。

### 三、实验方法

（1）彩色图像 RGB 转化为灰度图 Gray，opencv 上需要注意颜色空间是 RGB 还是 BGR，Cimg 中 RGB 分别对应 0,1,2 通道。

（2）因为霍夫圆检测对噪声比较敏感，所以首先要对图像做中值滤波（或者高斯滤波）去噪，平滑图像，消除图像噪声。

（3）图像边缘提取（梯度算子、拉普拉斯算子、canny 边缘检测算法）

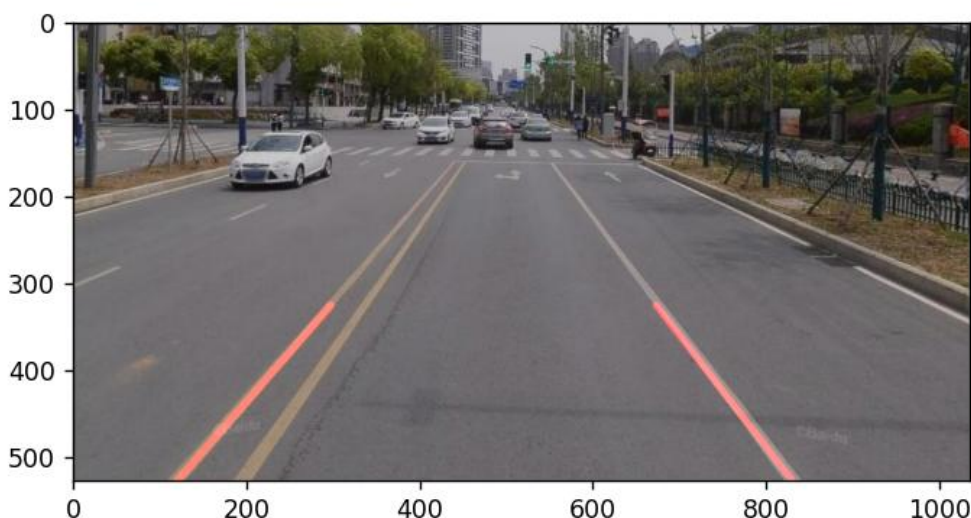
（4）图像二值化（判断此处是否为边缘点，就看灰度值==255),在高斯去噪和边界提取之后都需要二值化。

（5）映射到霍夫空间（此处准备两个容器，一个 Cimg 用来展示 hough-space 概况，一个数组 hough-space 用来储存 voting 的值，因为投票过程往往有某个极大值超过 255，多达几千，不能直接用灰度图来记录投票信息）。

（6）取局部极大值，设定阈值，过滤低于阈值的像素，排除干扰直线

（7）绘制直线。

### 四、实验结果



可以看到，由霍夫变换投票后，成功的拟合了车道线。

## 五、实验体会

在本次实验中，我使用 Python 语言结合 OpenCV、NumPy 和 Matplotlib 库来实现车道线检测的目的。实验流程主要包括图像的预处理、区域兴趣(ROI)的提取、霍夫变换进行直线检测、车道线绘制以及图像融合与显示。

具体参数设置包括高斯模糊核大小为 5，Canny 边缘检测的低高阈值分别为 50 和 150，霍夫变换参数  $\rho$  为 1， $\theta$  为  $\pi/180$ ，阈值为 15，最短线长为 40，最大线间隔为 20，实验成功地从图像中提取并显示了车道线。

实验过程也揭示了算法在弯道、不同光照条件或道路标记不清晰的情况下可能遇到的挑战。为了进一步提高车道线检测的准确性和稳定性，未来的工作可以考虑结合机器学习方法，并对参数进行自动调整以适应不同的道路和光照条件。这次实验不仅加深了我对车道线检测技术的理解，也为我在图像处理领域的学习提供了宝贵的实践经验。

## 课程实验三：学号识别

### 一、实验目的：

实验内容：手写数字的识别是机器视觉的入门级项目，是机器视觉的“Hello word”，其在实际场景中有广泛的应用场景。请设计手写数字识别方法识别自己的学号照片。

具体要求：

➤ 任务输入：学号照片。

- 任务输出：学号。
- 训练集：MNIST。
- 代码语言不限，方法不限，要求提交整个算法源代码，模型结果，算法分析等内容。
- 加分项（5分）：使用深度学习方法，代码环境名称以姓名缩写命名（例如吴晶晶的环境名：wj），实验报告中介绍代码环境配置过程。

## 二、实验原理

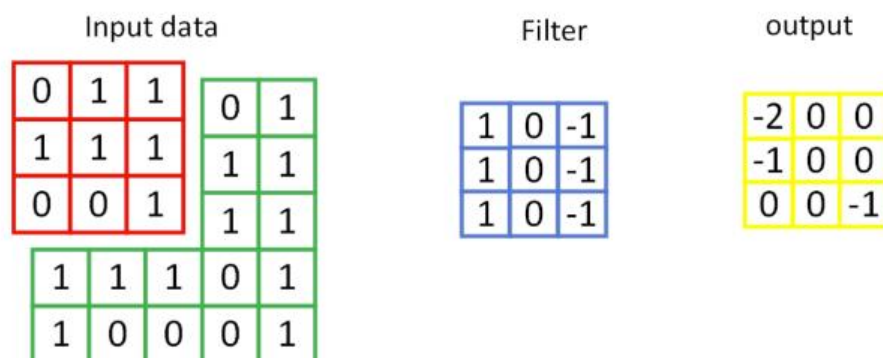
本次实验涉及的主要原理有：CNN 模型的构建、训练、识别，图像的分割。

### 2.1 构建模型 CNN

#### 1. 卷积层

每一个卷积核的通道数量要求和输入通道数量一样，卷积核的总数和输出通道的数量一样。

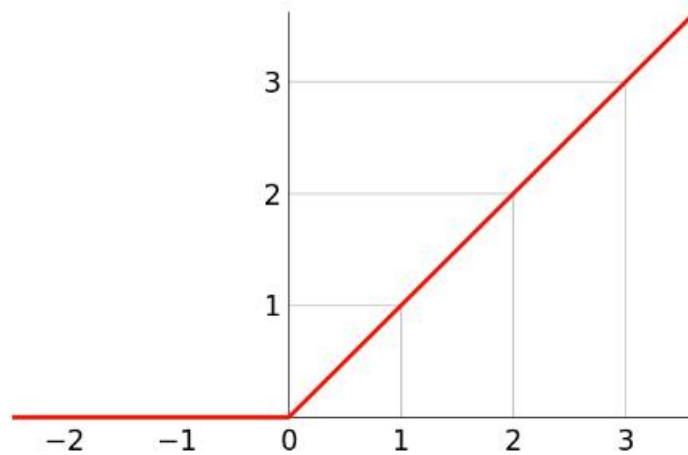
卷积(convolution)后，C(Channels)变，W(width)和 H(Height)可变可不变，取决于填充 padding。



#### 2. 激活层

激活层使用 ReLU 激活函数。

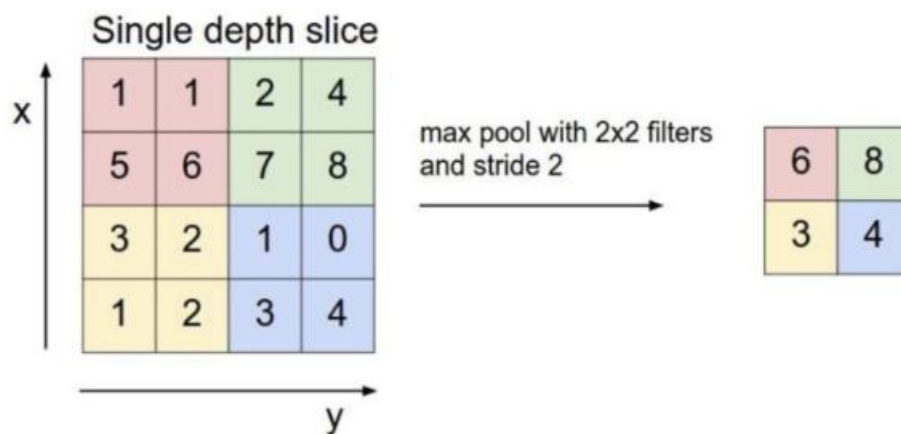
线性整流函数（Rectified Linear Unit, ReLU），又称修正线性单元，是一种人工神经网络中常用的激活函数（activation function），通常指代以斜坡函数及其变种为代表的非线性函数。



### 3.池化层

池化层采用最大池化。

池化(pooling)后，C(channels)不变，W(width)和 H(Height)变。



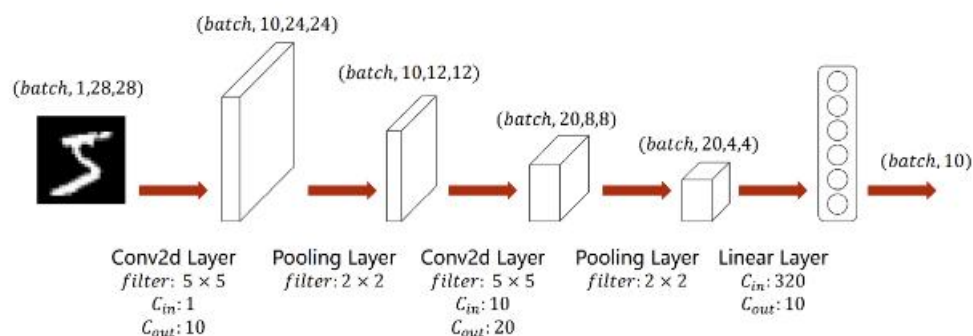
### 4.全连接层

之前卷积层要求输入输出是四维张量(B,C,W,H)，而全连接层的输入与输出都是二维张量(B,Input\_feature)，经过卷积、激活、池化后，使用 view 打平，进入全连接层。

### 5. CNN 模型

模型如图所示：





比如输入一个手写数字“5”的图像，它的维度为（batch,1,28,28）即单通道高宽分别为 28 像素。

首先通过一个卷积核为  $5 \times 5$  的卷积层，其通道数从 1 变为 10，高宽分别为 24 像素；

然后通过一个卷积核为  $2 \times 2$  的最大池化层，通道数不变，高宽变为一半，即维度变成（batch,10,12,12）；

然后再通过一个卷积核为  $5 \times 5$  的卷积层，其通道数从 10 变为 20，高宽分别为 8 像素；

再通过一个卷积核为  $2 \times 2$  的最大池化层，通道数不变，高宽变为一半，即维度变成（batch,20,4,4）；

之后将其 view 展平，使其维度变为 320(2044)之后进入全连接层，用线性函数将其输出为 10 类，即“0-9”10 个数字。

## 2.2 分割图像

本次实验我使用的分割方法是固定尺寸图像分割，鉴于此次图像分割只是将学号分隔开，很适合这种分割方法。

这里的原理是将图像均匀地切割成多个小部分，每部分包含单个数字。具体步骤如下：

### 1.图像预处理：

首先，图像被转换为灰度图像，并进行了反色处理，以使数字具有较高的对比度。

然后，图像被调整为特定的尺寸，这里是宽度为 280 像素，高度为 28 像素。

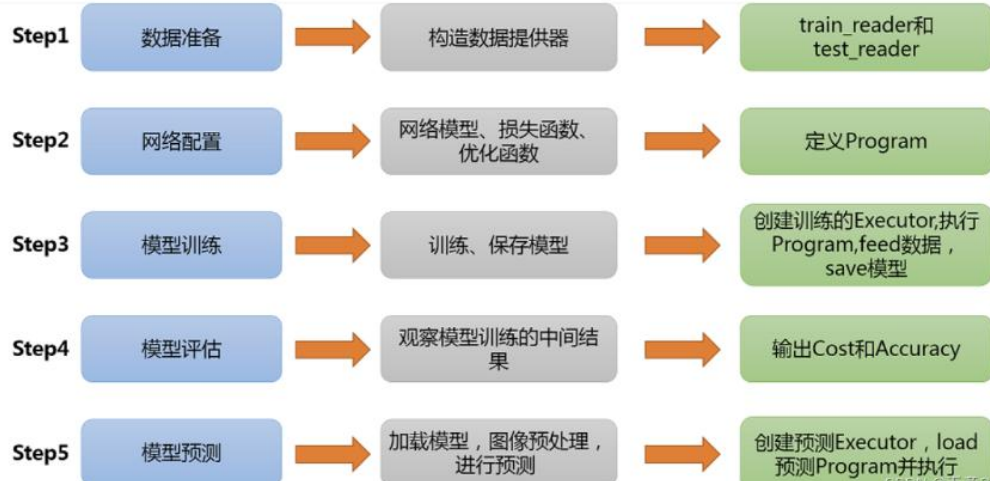
### 2.固定尺寸分割：

将处理后的图像分割为 10 个大小相等的部分，每部分宽度为 28 像素，高度为 28 像素（因为原图宽度是 280 像素）。

这种分割方式假设每个数字在图像中占据了相同的空间（即每个数字的宽度都是 28 像素）。

## 三、实验方法

构建好模型后，我们基于如下步骤进行模型训练与识别：



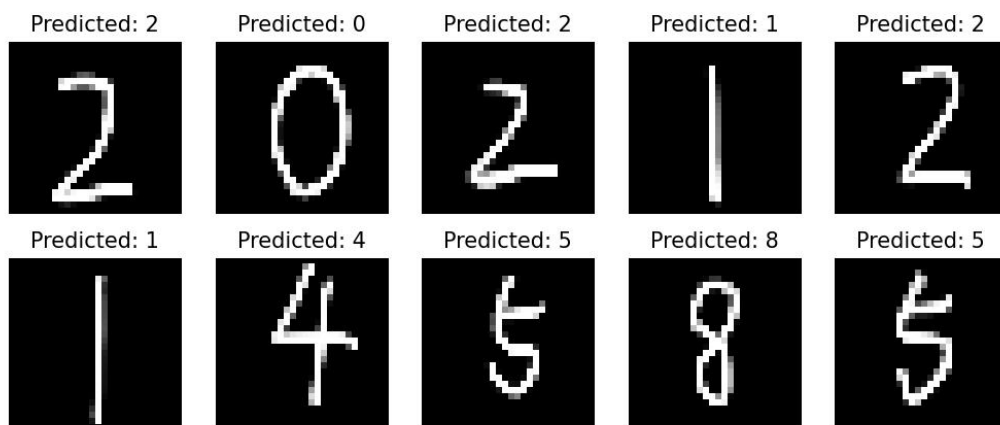
在具体的识别任务中，我们先使用分割算法将学号分割成类似手写识别训练集格式的图像，然后使用训练好的模型去进行手写数字识别。

#### 四、实验结果

输入图像：

2 0 2 1 2 1 4 5 8 5

输出识别结果：



#### 五、实验体会

在这次实验中，我深入探索了使用深度学习技术进行手写数字识别的过程，目标是识别学号照片中的数字。我选择了广为人知的 MNIST 手写数字数据集作为训练基础。整个实验流程分为几个关键步骤：数据预处理、模型设计、训练和测试，以及最终的学号识别应用。

在数据预处理阶段，我对 **MNIST** 数据集进行了加载和标准化处理，确保数据适合进行高效的深度学习。这个步骤对于提高模型的训练效率至关重要。

在训练阶段设计了一个卷积神经网络（**CNN**），它包含两个卷积层，每层之后都跟着 **ReLU** 激活函数和最大池化层。这种结构有助于提取图像的关键特征，同时保持计算的可管理性。通过这种方式，我能够构建一个既高效又有效的模型。

在训练阶段，我采用了交叉熵损失函数和随机梯度下降优化器来训练我的 **CNN** 模型。我仔细监控了每个训练周期的损失和准确率，确保模型在学习过程中持续改进。通过定期在测试集上评估模型，我能够跟踪其性能，并根据需要进行调整。

在模型训练完成后，将其应用于实际的学号识别任务。这涉及到对学号照片的预处理，包括灰度转换、大小调整和图像分割，以适应模型的输入要求。经过这些步骤后，模型能够成功地识别出照片中的数字。

整个实验过程不仅加深了我对深度学习在图像识别中应用的理解，还让我体会到了其在解决实际问题中的巨大潜力。通过这次实验，我学习到了如何处理和分析图像数据，以及如何设计和优化神经网络模型，这些技能对于我的未来学习和研究将非常有价值。