

# 第1讲--目标检测

---

## 主要内容

- ✓ 目标检测--目标分类和目标定位
- ✓ Deformable Parts Models
- ✓ HOG (Histogram Oriented Gradients)

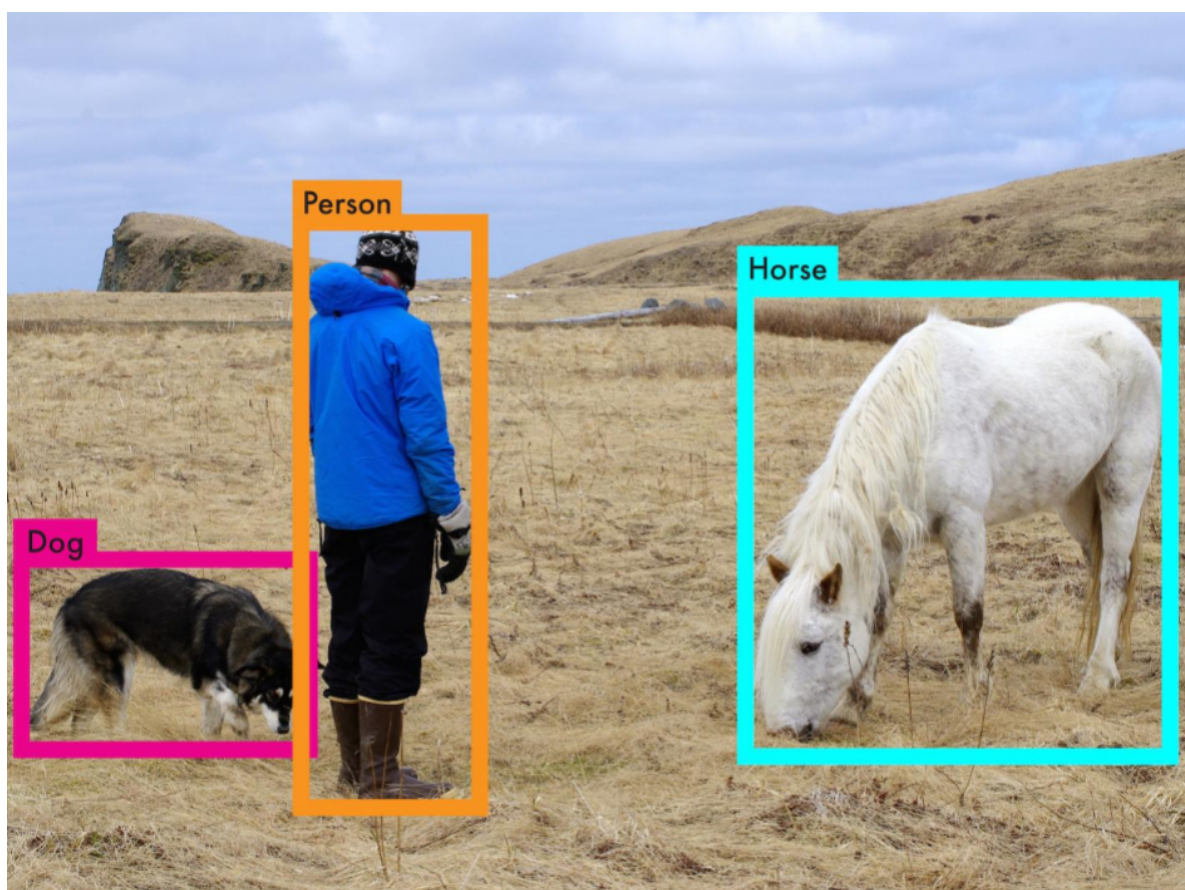
## 原理、代码实现

- ✓ 微积分-梯度（水平梯度的代码实现）
- ✓ Deformable Parts Models
- ✓ HOG (Histogram Oriented Gradients)

## 1 目标检测

---

目标检测主要包括目标的定位和分类。其中目标定位主要是将目标的出现范围用Bound boxing标记出来，分类主要是将目标进行归类。



## 2 目标检测论文

---

- ✓ P.F.Felzenszwalb, R.B.Girshick, D.McAllester, and D.Ramanan. Object detection with discriminatively trained part based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627-1645, 2010
- ✓ P.F.Felzenszwalb提出的可变性检测模型（Deformable parts models）。(代码的网址是<http://www.rossgirshick.info/latent/>)

## 3 Deformable Parts Models

- ✓ Deformable Parts Models可以在unix/linux/mac上运行。
- ✓ 前几年的Pascal Voc竞赛的Object Detection冠军都是采用这个框架的，并在这个框架的基础上进行修改的。现在大部分都用深度学习做目标检测。
- ✓ 在Windows环境下编译的需要安装Visual Studio集成开发环境

1. dt.cc文件中添加

```
#define int32_t int
```

2. 在features.cc 和 resize.cc中添加

```
#define bzero(a, b) memset(a, 0, b)
int round(float a)
{ float tmp = a - (int)a; if( tmp >= 0.5 ) return (int)a + 1; else return (int)a; }
```

3. 在resize.cc中:

```
alphainfo ofs[len]; 这句改成: alphainfo *ofs = new alphainfo[len]; 当然在同一作用域后面加上: delete []ofs
```

4. compile.m结尾改为

```
mex -O resize.cc
mex -O dt.cc
mex -O features.cc
mex -O getdetections.cc

% use one of the following depending on your setup
% 0 is fastest, 3 is slowest

% 0) multithreaded convolution using SSE
% mex -O fconvsse.cc -o fconv

% 1) multithreaded convolution using blas
%   WARNING: the blas version does not work with matlab >= 2010b
%   and Intel CPUs
% mex -O fconvblasMT.cc -lmwblas -o fconv

% 2) mulththreaded convolution without blas
% mex -O fconvMT.cc -o fconv

% 3) convolution using blas
% mex -O fconvblas.cc -lmwblas -o fconv

% 4) basic convolution, very compatible
% mex -O fconv.cc -o fconv
mex -O fconv.cc
```

5. 其他几个fconv用了其他平台的multiThread，在windows跑不起，改了上边的几个地方后，就可以运行了。跑demo.m看效果吧。



## 4 Python环境配置

- ☑ 安装Virtualenv
- ☑ Python的Virtualenv是不能修改Python.exe文件名称

```
pip3 install virtualenv
cd d:\
mkdir python37env
D:\python37env>virtualenv py37env
D:\python37env>cd py37env
D:\python37env\py37env>cd Scripts
D:\python37env\py37env\Scripts>activate //进入py37env虚拟环境

(py37env) D:\python37env\py37env\Scripts>pip list
Package      Version
-----
pip          20.3.1
setuptools   51.0.0
wheel        0.36.1
WARNING: You are using pip version 20.3.1; however, version 20.3.3 is available.
You should consider upgrading via the 'D:\python37env\py37env\Scripts\python.exe
-m pip install --upgrade pip' command.

(py37env) D:\python37env\py37env\Scripts>pip install ipython

(py37env) D:\python37env\py37env\Scripts>pip install opencv_python-4.1.2-cp37-
cp37m-win_amd64.whl

(py37env) D:\python37env\py37env\Scripts>pip uninstall numpy #删除numpy的新版本

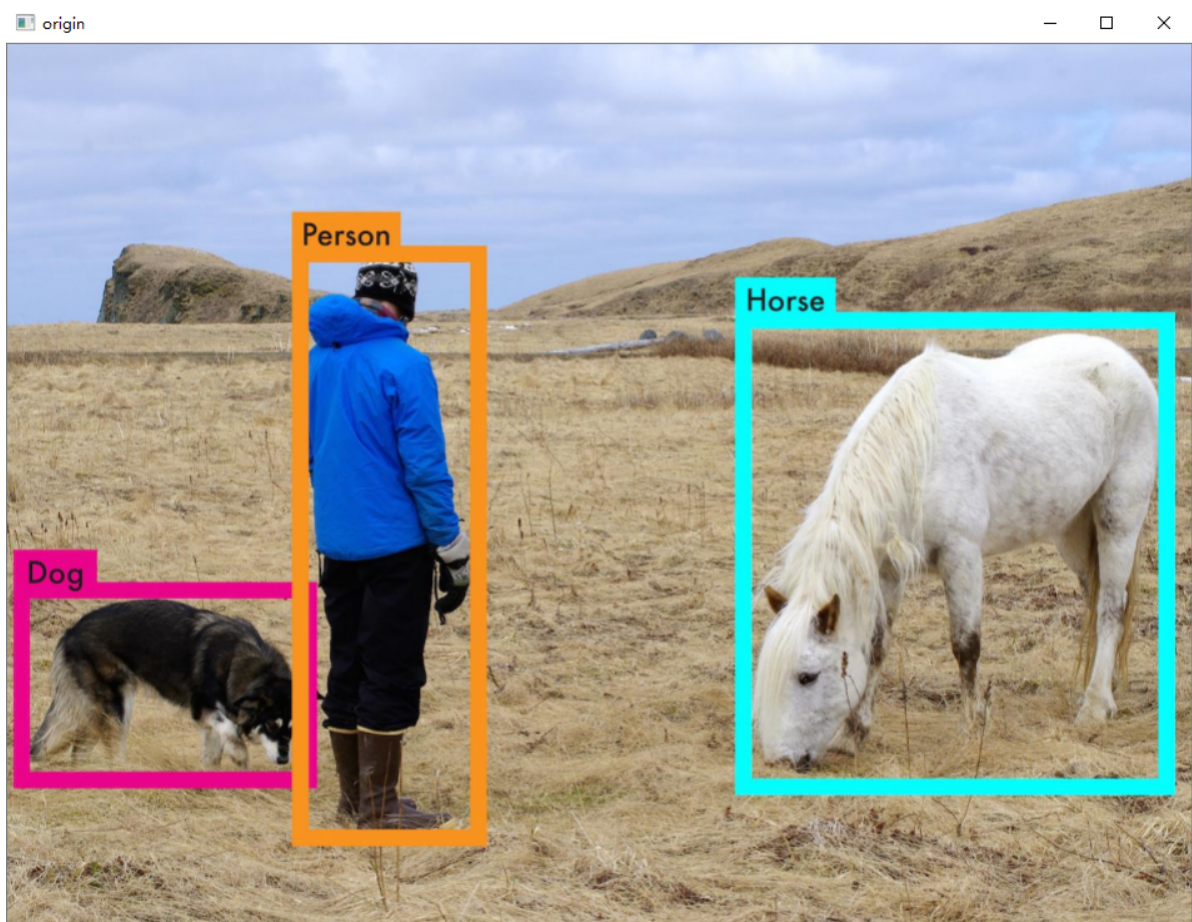
(py37env) D:\python37env\py37env\Scripts>pip install numpy=1.19.3 #安装numpy旧版本
```

```
(py37env) D:\python37env\py37env\Scripts>pip install scikit-image -i  
https://pypi.tuna.tsinghua.edu.cn/simple #安装scikit-image
```

## 4.1 通过OpenCV进行图像处理

✓ 将以下代码保存在py文件中，通过python.exe code1.py命令行执行，显示图像

```
import os  
import numpy as np  
import cv2  
os.chdir('F:/optimization/Markdown/fig1')  
img = cv2.imread('fig1.png',cv2.IMREAD_COLOR)  
cv2.imshow('origin',img)  
cv2.waitKey()==ord('q')
```



## 4.2 梯度和Sobel算子

✓ 梯度

$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

✓ 梯度的Magnitude

$$g = \sqrt{g_x^2 + g_y^2}$$

✓ 梯度的Direction



$$\theta = \arctan \frac{g_y}{g_x}$$

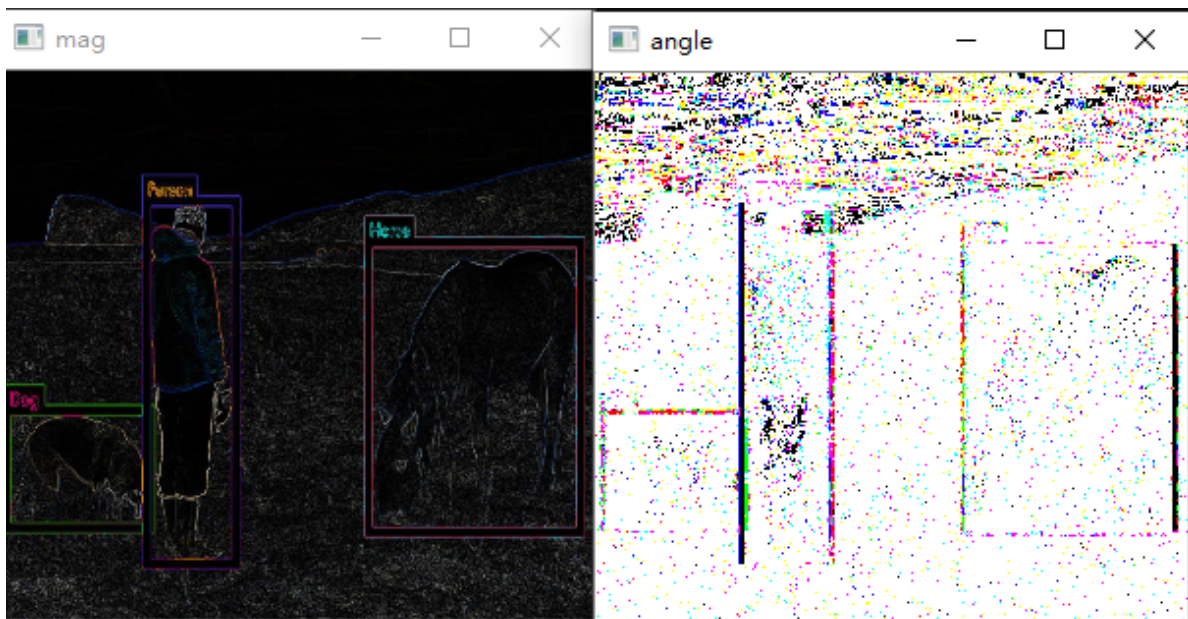
☑ 水平Sobel算子

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

☑ 垂直Sobel算子

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

```
import os
import numpy as np
import cv2
os.chdir('F:/optimization/Markdown/fig1')
image = cv2.imread('fig1.png', cv2.IMREAD_COLOR)
image = np.float32(image)/255.0
# Calculate gradient
gx = cv2.Sobel(image, cv2.CV_32F, 1, 0, ksize=1)
gy = cv2.Sobel(image, cv2.CV_32F, 0, 1, ksize=1)
mag, angle = cv2.cartToPolar(gx, gy, angleInDegrees=True)
cv2.namedWindow("mag", cv2.WINDOW_NORMAL)
cv2.imshow('mag', mag)
cv2.namedWindow("angle", cv2.WINDOW_NORMAL)
cv2.imshow('angle', angle)
cv2.waitKey(0) == ord('q')
```



☑ 梯度直方图

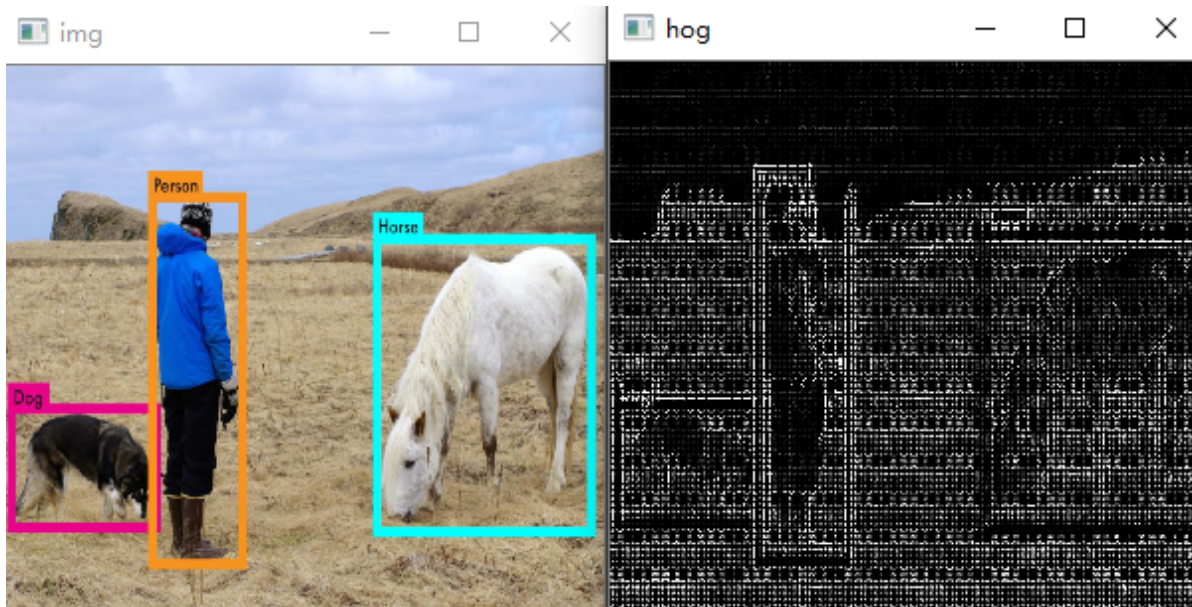
$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

```
from skimage import feature, exposure
from matplotlib import pyplot as plt
import os
import numpy as np
import cv2
```

```

os.chdir('F:/optimization/Markdown/fig1')
image = cv2.imread('fig1.png',cv2.IMREAD_COLOR)
fd, hog_image = feature.hog(image, orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(2, 4), visualize=True)
hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))
cv2.namedWindow("img",cv2.WINDOW_NORMAL)
cv2.imshow('img', image)
cv2.namedWindow("hog",cv2.WINDOW_NORMAL)
cv2.imshow('hog', hog_image_rescaled)
cv2.waitKey(0)==ord('q') #q键退出

```



### 4.3 自定义Sobel算子函数

- ✓ 自定义水平、垂直Sobel算子

```

from skimage import feature, exposure
from matplotlib import pyplot as plt
import os
import numpy as np
import cv2

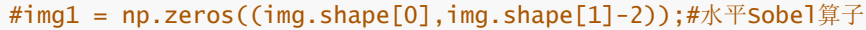

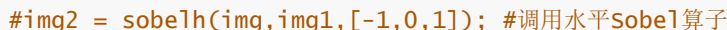

def sobelh(img,imgb, kernel):
    for i in range(img.shape[0]):
        for j in range(img.shape[1]-2):
            for k in range(len(kernel)):
                imgb[i,j] = imgb[i,j] + img[i,j+k] * kernel[k];
    return imgb;

def sobelv(img,imgb, kernel):
    for i in range(img.shape[0]-2):
        for j in range(img.shape[1]):
            for k in range(len(kernel)):
                imgb[i,j] = imgb[i,j] + img[i+k,j] * kernel[k];
    return imgb;

if __name__=="__main__":
    os.chdir('F:/optimization/Markdown/fig1');
    image = cv2.imread('fig1.png',cv2.IMREAD_COLOR);
    image = cv2.cvtColor(image,cv2.COLOR_BGR2HSV);

```

```

img = image[:, :, 0];
#水平Sobel算子
img1 = np.zeros((img.shape[0]-2, img.shape[1])); #垂直Sobel算子
#调用水平Sobel算子
img2 = sobelv(img, img1, [-1, 0, 1]); #调用垂直Sobel算子
cv2.namedWindow("img2", cv2.WINDOW_NORMAL);
cv2.imshow('img2', img2);
cv2.waitKey(0) == ord('q');

```



## 4.4 图像的Padding和梯度大小计算

```

from skimage import feature, exposure
from matplotlib import pyplot as plt
import os
import numpy as np
import cv2

def sobelh(img, imgb, kernel):
    for i in range(img.shape[0]):
        for j in range(img.shape[1]-2):
            for k in range(len(kernel)):
                imgb[i, j] = imgb[i, j] + img[i, j+k] * kernel[k];
    return imgb;

def sobelv(img, imgb, kernel):
    for i in range(img.shape[0]-2):
        for j in range(img.shape[1]):
            for k in range(len(kernel)):
                imgb[i, j] = imgb[i, j] + img[i+k, j] * kernel[k];
    return imgb;

def gm(img1, img2):
    img = img1 * img2;
    return np.sqrt(img);

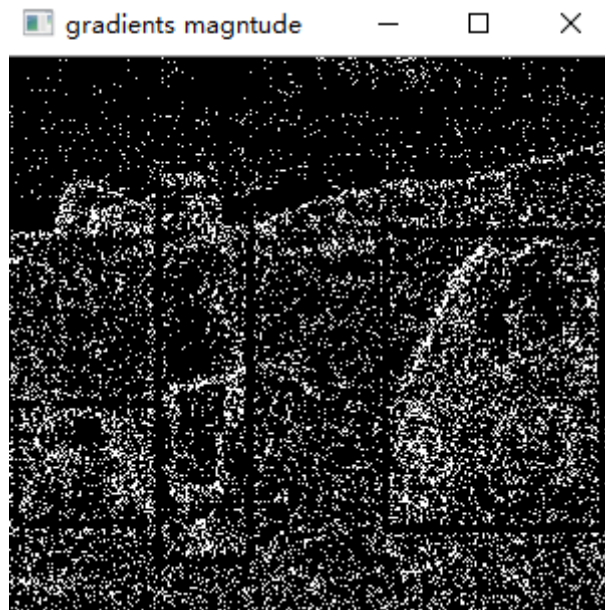
if __name__ == "__main__":
    os.chdir('F:/optimization/Markdown/fig1');
    image = cv2.imread('fig1.png', cv2.IMREAD_COLOR);
    image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV);
    img = image[:, :, 0];
    imgh1 = np.zeros((img.shape[0], img.shape[1]-2)); #水平Sobel算子

```

```

imgv1 = np.zeros((img.shape[0]-2,img.shape[1])); #垂直Sobel算子
imggh2 = sobelh(img,imgh1,[-1,0,1]); #调用水平Sobel算子
imgv2 = sobelv(img,imgv1,[-1,0,1]);
imggh2 = cv2.copyMakeBorder(imggh2,0,0,1,1,cv2.BORDER_REPLICATE) #上下方向扩充1个
像素
imgv2 = cv2.copyMakeBorder(imgv2,1,1,0,0,cv2.BORDER_REPLICATE) #左右方向扩充1个
像素
print(imggh2.shape);
print(imgv2.shape);
im = gm(imggh2,imgv2);
cv2.namedWindow("gradients magntude",cv2.WINDOW_NORMAL);
cv2.imshow('gradients magntude', im);
cv2.waitKey(0)==ord('q');

```



## 4.5 图像的梯度方向计算

```

def gd(img1,img2):
    '''
    computer the gradients directions
    '''
    img = img1/img2;
    return np.arctan(img);

def save_pickle(data,name):
    filename = name;
    f = open(filename,'wb');
    pickle.dump(data,f);
    f.close();

if __name__=="__main__":
    os.chdir('F:/optimization/Markdown/fig1');
    image = cv2.imread('fig1.png',cv2.IMREAD_COLOR);
    image = cv2.cvtColor(image,cv2.COLOR_BGR2HSV);
    img = image[:, :, 0];
    imggh1 = np.zeros((img.shape[0],img.shape[1]-2));#水平Sobel算子
    imgv1 = np.zeros((img.shape[0]-2,img.shape[1])); #垂直Sobel算子
    imggh2 = sobelh(img,imggh1,[-1,0,1]); #调用水平Sobel算子
    imgv2 = sobelv(img,imgv1,[-1,0,1]);

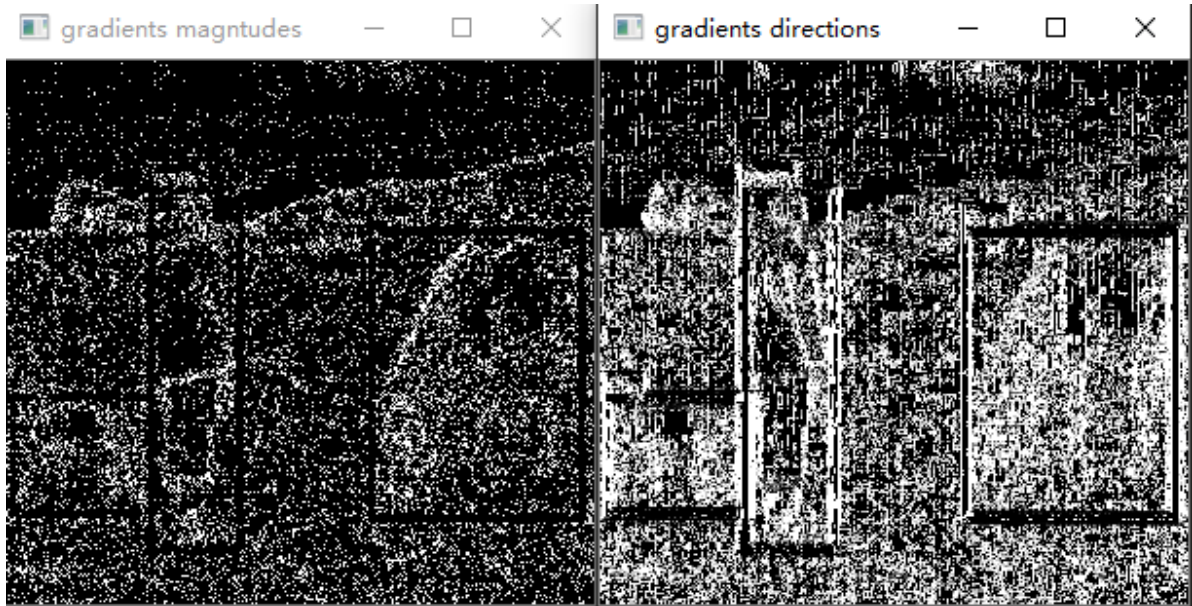
```



```

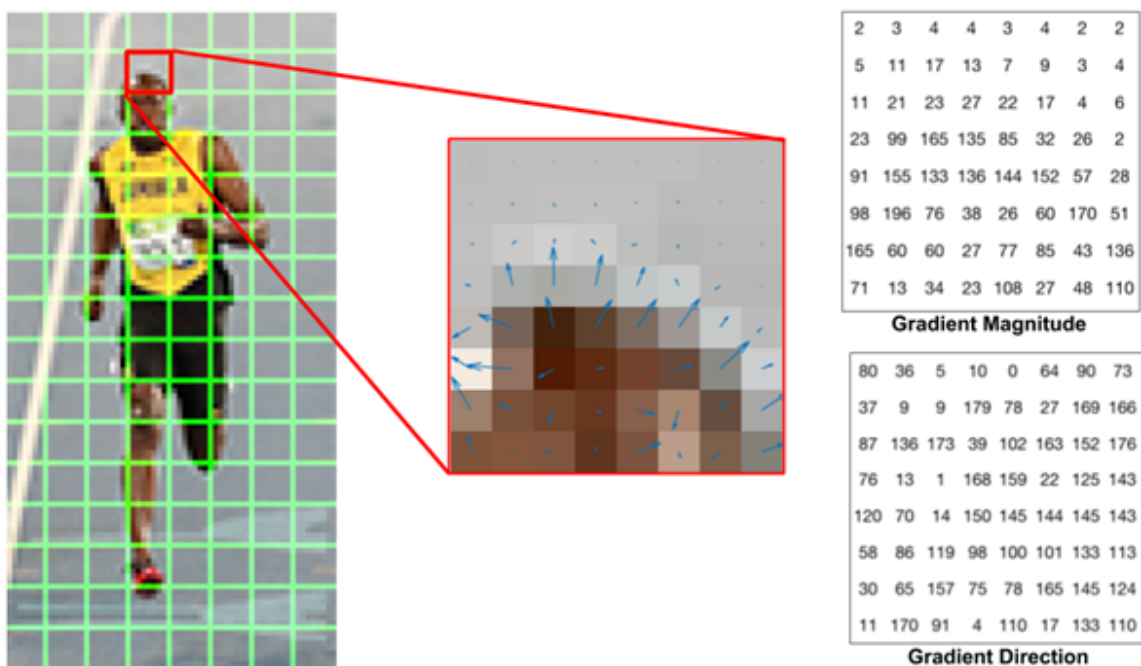
img_h2 = cv2.copyMakeBorder(img_h2,0,0,1,1,cv2.BORDER_REPLICATE) #上下方向扩充1个
像素
img_v2 = cv2.copyMakeBorder(img_v2,1,1,0,0,cv2.BORDER_REPLICATE) #左右方向扩充1个
像素
im1 = gm(img_h2,img_v2);
im2 = gd(img_h2,img_v2);
save_pickle(im1, 'F:/optimization/Markdown/code1/magnitude.pk');
save_pickle(im2, 'F:/optimization/Markdown/code1/direction.pk');
cv2.namedWindow("gradients magnitudes", cv2.WINDOW_NORMAL);
cv2.imshow('gradients magnitudes', im1);
cv2.namedWindow("gradients directions", cv2.WINDOW_NORMAL);
cv2.imshow('gradients directions', im2);
cv2.waitKey(0)==ord('q');

```

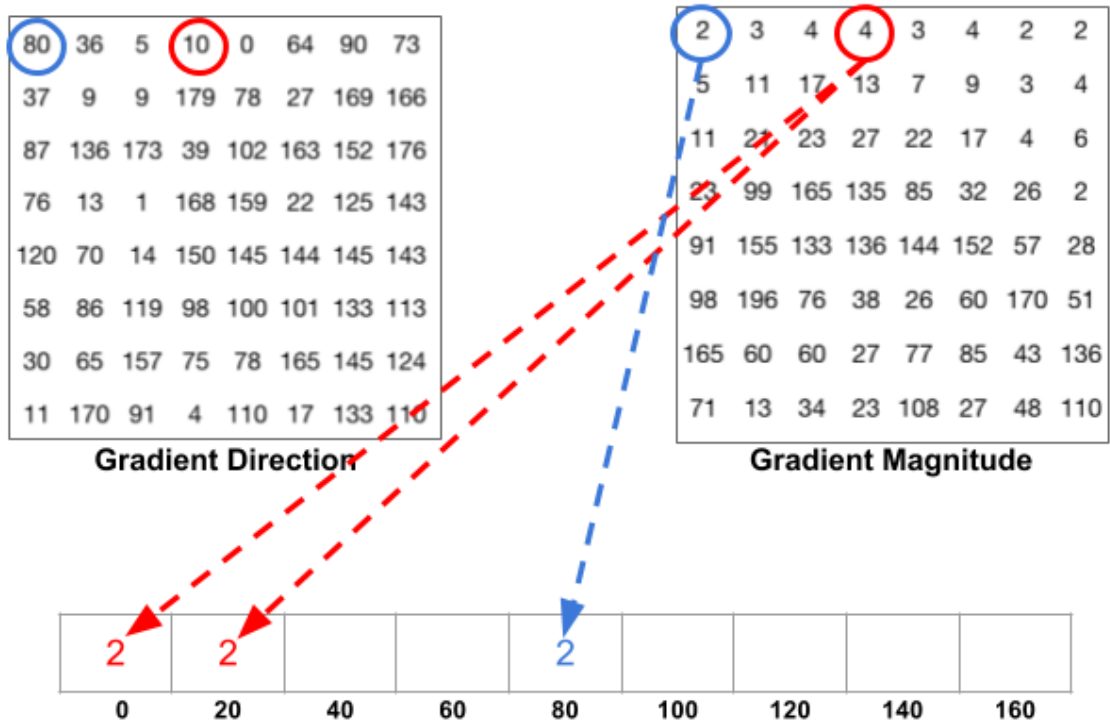


## 4.6 根据梯度的大小和方向构建直方图

☒ 获取图像的梯度大小和梯度方向



☒ 梯度方向作为横坐标，将梯度大小映射到角度区间



☑ 边界角度的处理

