实验内容:

1. 实践《视频处理》实验指导书中的视频处理相关代码，分析代码实现过

程中遇到的问题以及对实验结果进行分析。

2. 对于使用固定摄像头获取的视频文件，使用帧差法分析视频中的前景与

背景。

实现过程和结果:实现过程中因为电脑没有自带摄像头所以用视频代替

1. 从本地读取一段视频，并获取帧数，帧率以及时长。

导入需要的module

import cv2

import numpy as np

将视频相关参数存入cap变量中，获取帧数和帧率，并用公式计算出时长。

#从本地读取一段视频

cap=cv2.VideoCapture('./data/cat.avi')

#获取帧数，帧率以及时长

nbFrames = int(cap.get(cv2.CAP\_PROP\_FRAME\_COUNT))

fps = int(cap.get(cv2.CAP\_PROP\_FPS))

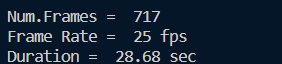
duration =(nbFrames/fps)

打印出结果。

print('Num.Frames = ' ,nbFrames)

print('Frame Rate = ',fps,'fps')

print('Duration = ', duration, 'sec')



1. 从摄像头读取视频并保存其中某一帧

导入opencv库

import cv2

捕捉摄像头（这里用视频替代）：

遍历每一帧，ret 判断视频是否正确读入，im 变量为实际读入视频帧。（cap.read returns a bool (True/False). If frame is read correctly, it will be True. So you can check end of the video by checking this return value.）

waitKey()等待用户按键：如果按下 esc 键（ascii 码 27），则退出应用，用户按下空格键，保存这一视频帧为jpg图片。

cap = cv2.VideoCapture('./data/cat.avi')

while True:

    ret, im = cap.read()

    cv2.imshow('video test', im)

    key = cv2.waitKey(10)

    if key == 27:

        break

    if key == ord(' '):

        cv2.imwrite('vid\_result.jpg',im)

cap.release()



最后释放资源和关闭窗口。

cap.release()

cv2.destroyAllWindows()

1. 在 opencv 窗口实时显示高斯模糊后的（彩色）图像

在上面代码基础上进行模糊处理

cap = cv2.VideoCapture('./data/cat.avi')

while True:

    ret, im = cap.read()

    if not ret:

        break

blur = cv2.GaussianBlur(im,(25,25),5)

    #src ==im ：要进行模糊的输入图像。

    #ksize == (25,25) ：高斯核的大小。它应该是一个奇数，比如 (3, 3)、(5, 5)、(7, 7) 等。这个值决定了模糊的程度。

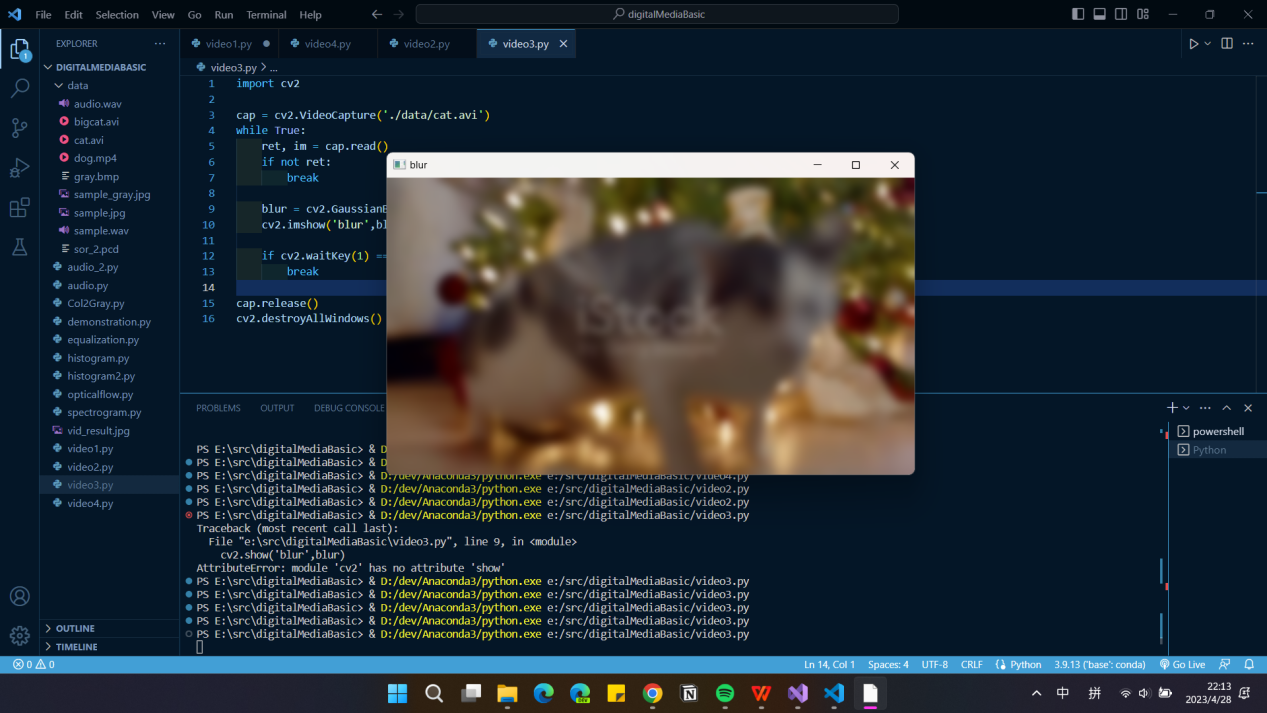
    #sigmaX =5 ：高斯核在 X 方向上的标准差。

    cv2.imshow('blur',blur)

    if cv2.waitKey(1) == 27:

        break

处理结果如下所示



1. 将视频读取到 NumPy 数组中

导入所需module

import cv2

import numpy as np

获取视频相关参数

cap = cv2.VideoCapture("./data/dog.mp4")

wid = int(cap.get(3))#CV\_CAP\_PROP\_FRAME\_WIDTH

hei = int(cap.get(4))#CV\_CAP\_PROP\_FRAME\_HEIGHT

framerate = int(cap.get(5))#cv2.CAP\_PROP\_FPS

framenum = int(cap.get(7))#cv2.CAP\_PROP\_FRAME\_COUNT

创建存储视频的数组，以每帧为单位存储，大小为framenum\*hei\*wid\*3（帧数\*每帧的像素信息），cnt用来记录数组的元素个数，初始化为0

video = np.zeros((framenum,hei,wid,3),dtype='float16')

cnt = 0

对每一帧进行捕获，显示并存入数组，a为bool值判断每一帧是否正常读入，b为储存帧的变量。

while(cap.isOpened()):

    a, b = cap.read()

    if a:

        cv2.imshow('%d'%cnt, b)

        cv2.waitKey(20)

        b = b.astype('float16')/255

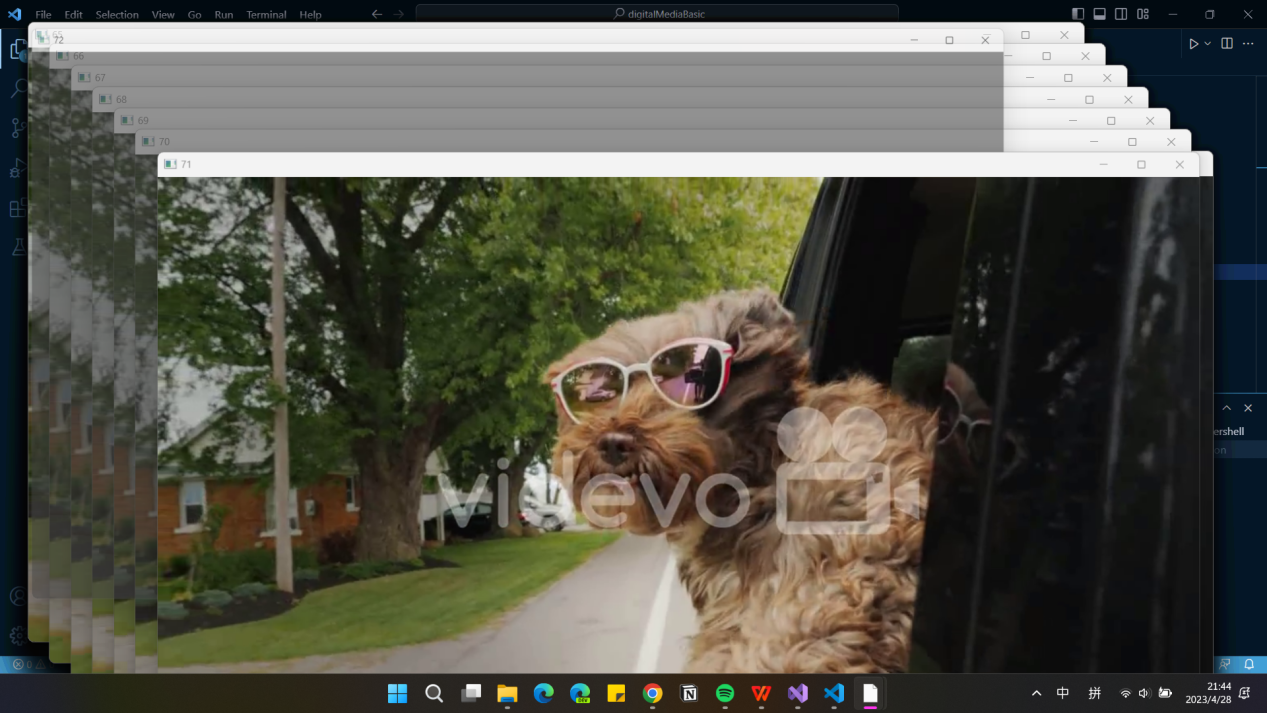
        video[cnt]=b

        cnt += 1

    else:

        break

运行结果如下：



1. 使用固定摄像头获取的视频文件，使用帧差法分析视频中的前景与背景。

因为运动目标与背景的差异是在视频帧之间出现的，因此使用前后两帧之间的差异来进行运动目标的检测，从而实现背景的建模和前景的提取。读取视频文件并初始化第一帧作为背景帧，对每一帧进行灰度转换和高斯模糊操作来减少噪声的干扰，计算当前帧和前一帧之间的差异，将帧间差异图像进行二值化处理，将背景和前景区分开来。接着为了更便于分析，对二值图像进行形态学操作，去除噪声和填充前景区域内部空洞。通过在原始帧上绘制前景轮廓框，识别并显示出视频中的运动目标。具体代码如下：

import cv2

# 打开视频文件

cap = cv2.VideoCapture('./data/carflow.mp4')

# 读取第一帧

ret, frame1 = cap.read()

# 对第一帧进行灰度转换和高斯模糊

gray1 = cv2.cvtColor(frame1, cv2.COLOR\_BGR2GRAY)

gray1 = cv2.GaussianBlur(gray1, (21, 21), 0)

# 开始循环

while True:

    # 读取下一帧

    ret, frame2 = cap.read()

    # 判断是否到达视频末尾

    if not ret:

        break

    # 对当前帧进行灰度转换和高斯模糊

    gray2 = cv2.cvtColor(frame2, cv2.COLOR\_BGR2GRAY)

    gray2 = cv2.GaussianBlur(gray2, (21, 21), 0)

    # 计算当前帧与前一帧的差异

    diff = cv2.absdiff(gray1, gray2)

    # 对差异图像进行二值化处理

    thresh = cv2.threshold(diff, 25, 255, cv2.THRESH\_BINARY)[1]

    # 去除二值图像中的噪声

    thresh = cv2.dilate(thresh, None, iterations=2)

    thresh = cv2.erode(thresh, None, iterations=2)

    # 在原始帧上绘制前景的轮廓

    contours, hierarchy = cv2.findContours(thresh.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

    for contour in contours:

        if cv2.contourArea(contour) < 1000:

            continue

        (x, y, w, h) = cv2.boundingRect(contour)

        cv2.rectangle(frame2, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # 创建一个窗口，大小为640x480

    cv2.namedWindow('frame', cv2.WINDOW\_NORMAL)

    cv2.resizeWindow('frame', 640, 480)

    cv2.namedWindow('thresh', cv2.WINDOW\_NORMAL)

    cv2.resizeWindow('thresh', 640, 480)

    # 显示原始帧和分析结果

    cv2.imshow('frame', frame2)

    cv2.imshow('thresh', thresh)

    # 更新前一帧

    gray1 = gray2.copy()

    # 按下 q 键退出循环

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# 释放视频资源和窗口

cap.release()

cv2.destroyAllWindows()

实现结果如图所示：

