

Opencv图像处理

导师: Zhang



deepshare.net

深度之眼

第五课：运动目标识别

运动目标识别

摄像头调用

视频的读取与保存

帧差法

光流法

背景减除法

1. 摄像头调用

开启摄像头

- 函数1: `cv2.VideoCapture()`
 - 参数说明: 0, 1代表电脑摄像头, 或视频文件路径。
- 函数2: `ret, frame = cap.read()`
- 说明: `cap.read()` 按帧读取视频,
 - Ret: 返回布尔值True/False, 如果读取帧是正确的则返回True, 如果文件读取到结尾, 它的返回值就为False;
 - Frame: 每一帧的图像, 是个三维矩阵。



开启摄像头

- 下面的程序将使用opencv调用摄像头，并实时播放摄像头中画面，按下“q”键结束播放

```
import numpy as np
import cv2
cap = cv2.VideoCapture(0)
while(True):
    #获取一帧帧图像
    ret, frame = cap.read()
    #转化为灰度图
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow('frame', gray)
    #按下“q”键停止
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

代码讲解 

2. 播放、保存视频

保存视频

- 指定写入视频帧编码格式
- 函数 `fourcc = cv2.VideoWriter_fourcc('M', 'J', 'P', 'G')`

OpenCV 4.1版本标志	OpenCV 4.0版本标志	作用
<code>VideoWriter::fourcc('D','I','V','X')</code>	<code>CV_FOURCC('D','I','V','X')</code>	MPEG-4编码
<code>VideoWriter::fourcc('P','I','M','1')</code>	<code>CV_FOURCC('P','I','M','1')</code>	MPEG-1编码
<code>VideoWriter::fourcc('M','J','P','G')</code>	<code>CV_FOURCC('M','J','P','G')</code>	JPEG编码（运行效果一般）
<code>VideoWriter::fourcc('M','P','4','2')</code>	<code>CV_FOURCC('M','P','4','2')</code>	MPEG-4.2编码
<code>VideoWriter::fourcc('D','I','V','3')</code>	<code>CV_FOURCC('D','I','V','3')</code>	MPEG-4.3编码
<code>VideoWriter::fourcc('U','2','6','3')</code>	<code>CV_FOURCC('U','2','6','3')</code>	H263编码
<code>VideoWriter::fourcc('I','2','6','3')</code>	<code>CV_FOURCC('I','2','6','3')</code>	H263I编码
<code>VideoWriter::fourcc('F','L','V','1')</code>	<code>CV_FOURCC('F','L','V','1')</code>	FLV1编码

保存视频

- 创建VideoWriter对象


- 函数out =

```
cv2.VideoWriter('output.avi', fourcc, 20.0,  
(640, 480))
```

- 参数说明:

- 参数1: 保存视频路径+名字;
- 参数2: FourCC 为4 字节码, 确定视频的编码格式;
- 参数3: 播放帧率
- 参数4: 大小
- 参数5: 默认为True, 彩色图

```
#调用摄像头函数cv2.VideoCapture, 参数0: 系统摄像头  
cap = cv2.VideoCapture(0)  
#创建编码方式  
fourcc = cv2.VideoWriter_fourcc('M', 'J', 'P', 'G')  
#创建VideoWriter对象  
out = cv2.VideoWriter('output.avi', fourcc, 20.0,  
                      (640, 480))
```

代码讲解 

3. 帧差法目标识别

帧差法

- 帧间差分法是通过视频中对相邻两帧图像做差分运算来标记运动物体的方法。
- 当视频中存在移动物体时，相邻帧（或相邻三帧）之间在灰度上会有差别，求取两帧图像灰度差的绝对值，则静止的物体在差值图像上表现出来全是0，而移动物体特别是移动物体的轮廓处由于存在灰度变化为非0。

帧差法

- 优点：
 - 算法实现简单，程序设计复杂度低；
 - 对光线等场景变化不太敏感，能够适应各种动态环境，稳定性较好；
- 缺点：
 - 不能提取出对象的完整区域，对象内部有“空洞”；
 - 只能提取出边界，边界轮廓比较粗，往往比实际物体要大；
 - 对快速运动的物体，容易出现糊影的现象，甚至会被检测为两个不同的运动物体；
 - 对慢速运动的物体，当物体在前后两帧中几乎完全重叠时，则检测不到物体；

4. 光流法

光流法

- 光流法利用图像序列中像素在时间域上的变化以及相邻帧之间的相关性，根据上一帧与当前帧之间的对应关系，计算得到相邻帧之间物体的运动信息。
- 大多数的光流计算方法计算量巨大，结构复杂，且易受光照、物体遮挡或图像噪声的影响，鲁棒性差，故一般不被对精度和实时性要求比较高的监控系统所采用

光流法

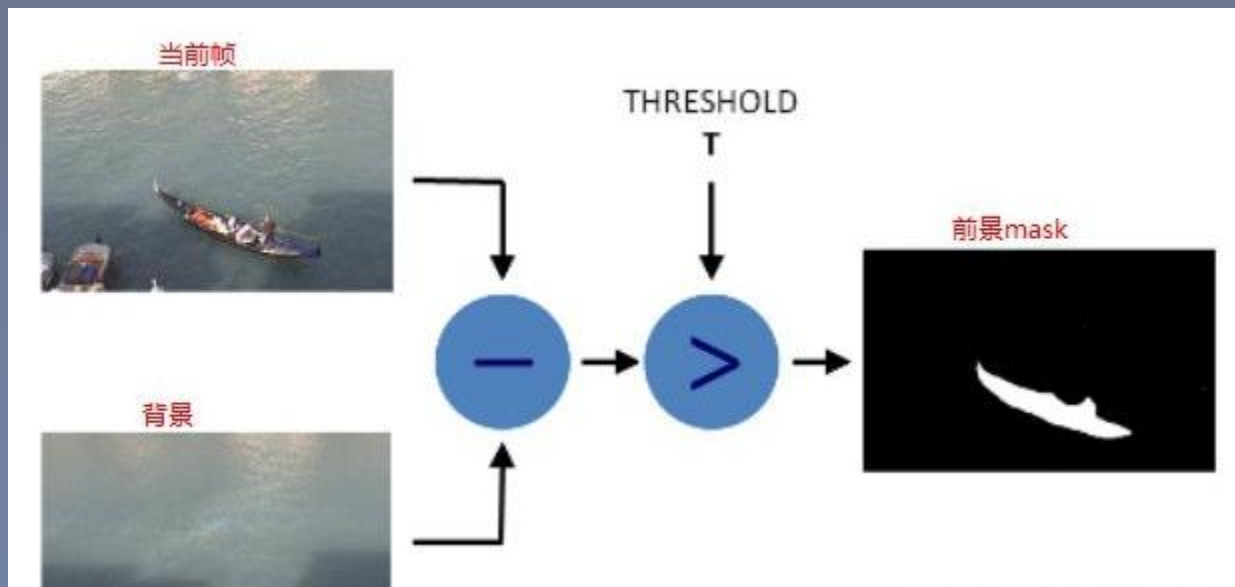
- 光流是基于以下假设的：
 - 在连续的两帧图像之间（目标对象的）像素的灰度值不改变。
 - 相邻的像素具有相同的运动

5. 背景减除法

背景减法

- 背景消除

- OpenCV中常用的两种背景消除方法，一种是基于高斯混合模型GMM实现的背景提取，另外一种是基于最近邻KNN实现的。



背景减除法

- GMM模型

- MOG2算法，高斯混合模型分离算法，它为每个像素选择适当数量的高斯分布，它可以更好地适应不同场景的照明变化等
- 函数：`cv2.createBackgroundSubtractorMOG2(int history = 500, double varThreshold = 16, bool detectShadows = true)`

- KNN模型

- `cv2.createBackgroundSubtractorKNN()`

背景减法

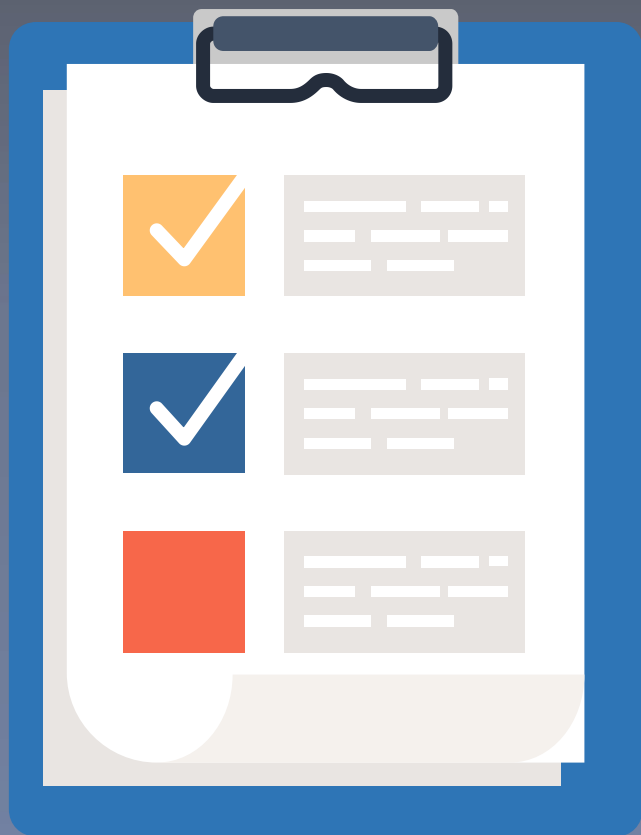
- 方法

- 主要通过视频中的背景进行建模，实现背景消除，生成mask图像，通过对mask二值图像分析实现对前景活动对象的区域的提取，整个步骤如下：

- 1. 初始化背景建模对象GMM
- 2. 读取视频一帧
- 3. 使用背景建模消除生成mask
- 4. 对mask进行轮廓分析提取ROI
- 5. 绘制ROI对象

代码讲解 

本课总结



核心知识

- ✓ opencv摄像头调用
- ✓ 保存视频，修改视频格式
- ✓ 基于视频的运动目标检测，掌握常见的三种方法：光流法/
帧差法/背景减除法的使用



重点 重点来了!

本节作业

Preview of next lesson



01 本节代码复现和理解



02 保存不同格式的视频，对比不同格式的区别

03 手机拍摄一段运动物体视频，使用某种方法进行目标检测



深度之眼
deepshare.net

联系我们：

电话：18001992849

邮箱：service@deepshare.net

Q Q：2677693114



公众号



客服微信

