任选一种方法（背景减法、帧差法等）编程实现视频中运动目标检测，并分析实验结果。

**实现方法**

1. **框架构建**

实验实现了一个简单的视频处理程序，使用了 OpenCV 库来读取视频并进行运动物体检测。

* 1. 打开指定路径的视频文件"exp2.avi"，并通过VideoCapture进行读取。
  2. 构建视频帧处理循环，在一个循环中通过VideoCapture进行读取每一帧并对每一帧进行处理。
  3. 对当前帧和前一帧进行了两种不同的运动物体检测：moveCheck2 使用背景减法，而 moveCheck1 使用帧差法进行检测。进行完当前帧的处理后，将当前帧复制给 preframe，以便下一次迭代时可以使用前一帧的图像进行处理。
  4. 同时检测用户按下键盘上的 'q' 键，如果按下则退出循环，结束程序运行。
  5. 循环部分代码如下：

while (cap.read(frame))

{

imshow("video", frame);

if (!preframe.empty())

{

moveCheck2(frame, pMOG2);//背景减法

moveCheck1(preframe, frame);//帧差法

}

frame.copyTo(preframe);//更新前一帧的参数

if (waitKey(10) == 'q') {

break;

}

}

总之该程序通过循环处理视频的每一帧，结合使用了背景减法器和帧差法来进行运动物体检测，并在图像中标记出检测到的物体。

1. **背景减法**
   1. 创建BackgroundSubtractorMOG2以及初始化

在主函数中，开始对每一帧进行循环之前，用Ptr<BackgroundSubtractor> pMOG2 = createBackgroundSubtractorMOG2(500, 100, false)语句创建了一个 MOG2 背景减法器对象，用于后续的运动物体检测。

createBackgroundSubtractorMOG2 是 OpenCV 中用于创建 MOG2 背景减法器的函数。设置参数解释如下：

MOG2 背景减法器是一种用于运动物体检测的算法，它基于混合高斯模型来对场景的背景进行建模，并通过比较当前帧与背景模型的差异来识别运动物体。因为背景减法需要背景图像，然而我们不可能手动得到每个视频的背景，opencv提供了 MOG2 背景减法器进行实时的检测背景，建立背景图从而进行后续的背景减法侦测运动对象。

在开始时，MOG2 背景减法器利用一定数量的历史帧（由参数指定）来建立场景的背景模型。在代码中，设置该参数为500，较大的值意味着更长的历史信息用于背景建模，生成通常情况下更稳定的背景估计，缺点是会增加计算成本。

MOG2 背景减法器使用多个高斯分布来建模每个像素的背景。这个参数指定了用于建模背景的高斯分布数量。参数100是混合高斯模型的数量，也被称为高斯分量数。更多的高斯分布通常能更好地适应不同场景下的背景变化，但可能会增加计算复杂度。

false 表示未启用阴影检测。对于启用了阴影检测的情况，MOG2 背景减法器还会尝试区分背景中的阴影部分。阴影通常会与前景有相似的颜色或亮度，但与其在场景中的位置不同。通过阴影检测算法，将被误识别为前景的阴影部分标记为不同于前景的对象。这里关闭阴影检测减小性能开销。

值得注意的是，创建BackgroundSubtractorMOG2需要在主函数中进行，如果在遍历每一帧的循环中创建会发生错误。

* 1. 背景减法处理

pMOG2->apply(frame, fgMask, 0.005)将当前帧 frame 与背景模型进行比较，生成前景掩码 fgMask，表示图像中变化的区域。

当新的帧进入时，背景减法器pMOG2会比较当前帧与背景模型的差异。对于每个像素，它将计算当前像素值与背景模型的高斯分布之间的差异。如果当前像素值与模型的某个高斯分布的差异超过了一个阈值，则将该像素标记为前景。

下图所示为应用了背景减法器之后的fgMask，可见整幅图像为二值图像，运动对象生成为前景，但是此时画面上依然有噪点和干扰物体（除了人之外的运动物体），同时还存在对于人的检测效果较差的情况（身体部位不全，分为好几个连通区域，或是多人构成一个连接区域）。



* 1. 去除噪点

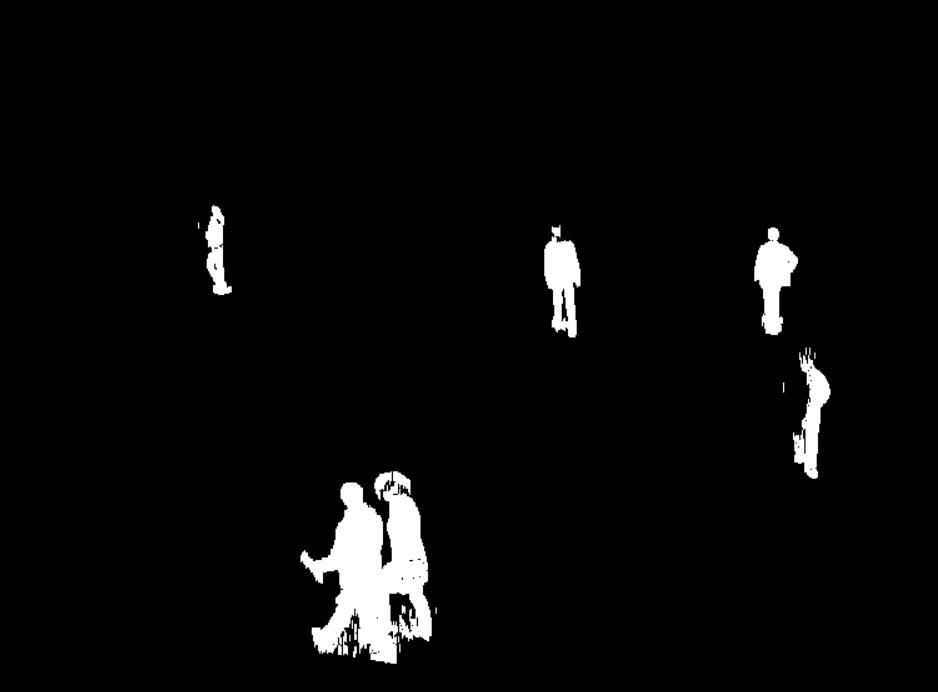
我们希望达到的理想效果是每个人只构成一个联通区域，且一个连通区域只代表一个人，另外排除掉前景中的噪点和无关物体。这里使用形态学的处理方式，用先腐蚀的方式先腐蚀掉较小的前景，去掉无关物体和噪点，并且分开多人构成的连通域，再进行膨胀操作，膨胀操作则有助于连接物体区域并填充空洞，使得每个人仅构成一个联通区域。可以用开操作来完成以上逻辑。

Mat line = getStructuringElement(MORPH\_RECT, Size(1, 5), Point(-1, -1));

morphologyEx(fgMask, fgMask, MORPH\_OPEN, line);

使用形态学运算去除前景掩码中的噪点，MORPH\_OPEN 表示开操作，通过 line 结构元素对图像进行处理。

经过开操作后的前景图像如下图所示，可以观察到噪点和无关物体已经消失，人形也更加完整，运动物体更加连续，证明这向我们理想的效果靠近了。



1. **帧差法**
   1. 转换为灰度图像：

cvtColor(preframe, preGray, COLOR\_BGR2GRAY);

cvtColor(frame, gray, COLOR\_BGR2GRAY);

将前一帧图像和当前帧图像转换为灰度图像，便于后边的操作。



* 1. 计算帧差图：

absdiff(preGray, gray, diff);

通过对前一帧和当前帧的灰度图像进行绝对差分，得到两帧之间差异图像 diff，差异图像上反映了可能是运动物体的地方。下图为其中一幅帧差图，可以看到显示的运动物体较为模糊，接下来还需要对其进行二值化和形态学的操作。



* 1. 二值化：

threshold(diff, binary, thresholdValue, 255, THRESH\_BINARY);

使用阈值化将差分图像转换为二值图像。这里使用了阈值25（手动调节）来进行二值化。

下图为二值化后的一帧图像，可以看到检测后运动物体比前面的帧差图更加清晰，但是仍然有无关物体和噪点在图像中，检测到的人体线条感较重，边缘和物体中心不完整不连续，我们需要进行形态学操作来处理以上问题。

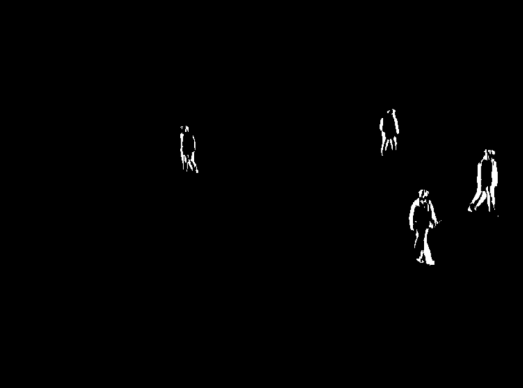
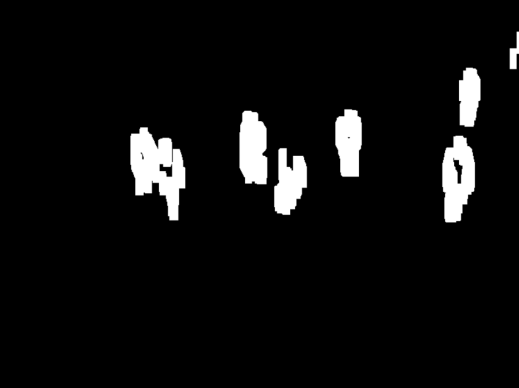


* 1. 腐蚀和膨胀：

erode(binary, binary, eroElement);

dilate(binary, binary, dilElement);

通过腐蚀和膨胀操作对二值图像进行形态学处理。和上面背景减法的逻辑相同，腐蚀操作有助于去除小的噪声点，而膨胀操作则有助于连接物体区域并填充空洞，使得运动物体更加连续。这里我们用较上面所用的开运算更大的模板进行膨胀操作，弱化人物的线条感，使其整体变为块状，整体更加连续一体。下面为经过形态学操作后得到的图像。

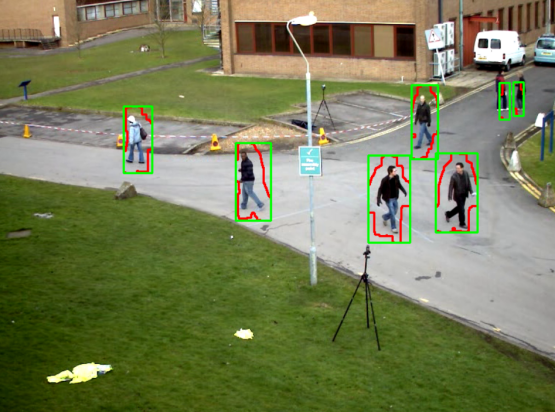
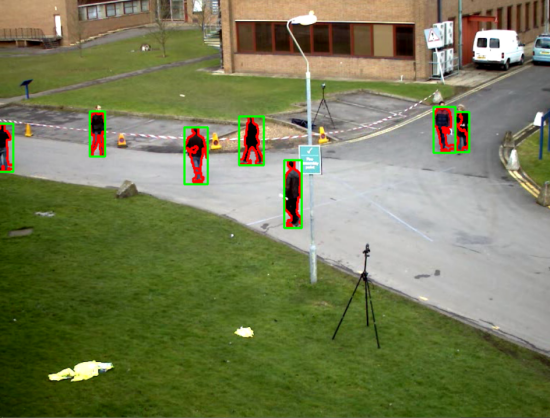
Erode Dilate

1. **绘制轮廓和矩形框**
   1. vector<vector<Point>> contours;

findContours(fgMask, contours, RETR\_EXTERNAL, CHAIN\_APPROX\_SIMPLE);

使用 findContours 找到前景掩码中的轮廓，存储在 contours 中。

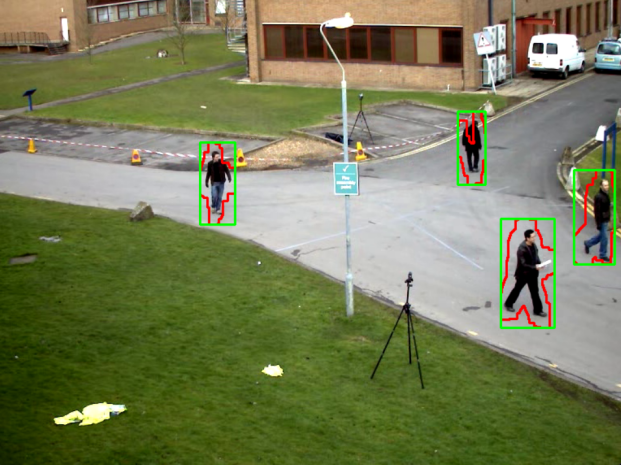
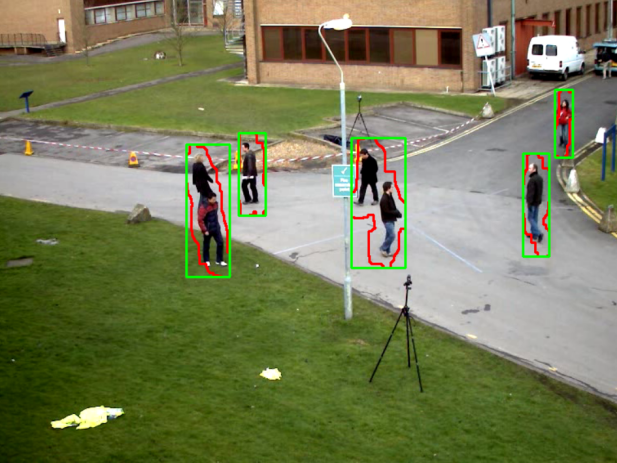
* 1. 对于每个轮廓：
     1. 计算轮廓的面积 area，若面积小于 150 （自设阈值）则跳过。这样可以减少前一步未腐蚀掉的噪点的影响。
     2. 计算轮廓的外接矩形 boundBox。
     3. 在原始图像 frame 上绘制检测到的轮廓和外接矩形。

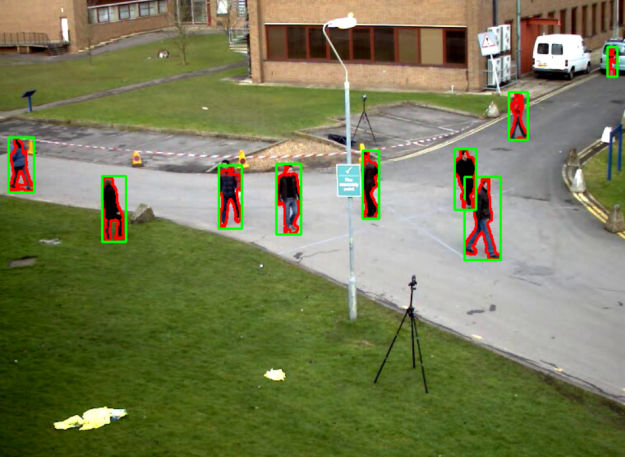
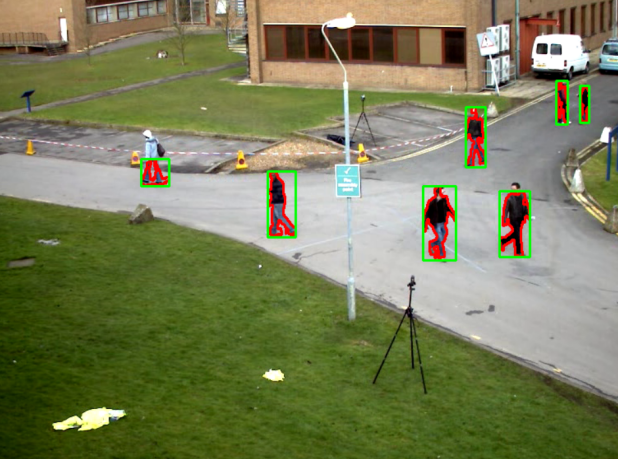
帧差法绘制轮廓 背景减法绘制轮廓

**实验结果**

下面为程序运行的部分截图。

帧差法1 帧差法2

背景减法1 背景减法2

**结论分析**

根据上面的实验结果可以看出，最后检验运动物体的关键是能否在二值图中将运动物体完整地分离出来。

在帧差法中，由于二值图像中运动物体块较小，所以要用更大的filter进行膨胀，这样能避免检测的运动对象显示轮廓（包围形）闪烁且不稳定，但是缺点是对多人的连通域不敏感。

在背景减法中，二值图像中的运动物块较为完整，因此可以进行较小filter的膨胀操作，这也导致了在本实验中的背景减法算法中对多人连通域的处理更好。下图为对多人的连通域的分离示意图。

多人的连通域未分离 多人的连通域分离

但同时，有少数情况，某些运动物体因为膨胀不够而不会被检测到。

除此以外，帧差法和背景减法各有优缺点：

帧差法简单直观，容易理解，只需对像素进行简单的差分运算，计算速度较快。但是对光照变化、噪点敏感，可能会将光照变化或背景细微变化误认为是运动物体，容易受到摄像头自身振动、噪声等因素的干扰，可能导致误检测。且对于背景长时间变化的场景，比如潮汐涨落或者光线变化很大的情况，帧差法的效果可能不理想。

背景减法能够根据场景自适应更新背景模型，对于背景变化较大的情况更具有鲁棒性，能够较好地处理光照变化和减少噪声影响。但是计算复杂度高，可能需要较多的计算资源。在某些情况下，可能由于背景模型不准确或者参数设置不合适而导致漏检或误检。

在实际应用中，我们可以需要根据具体场景和需求选择合适的方法或结合多种方法进行综合处理。