《数字图像处理》综合实验报告

姓名：陆逸航　　学号：U201716983　　班级：软工 1702 班

1 项目简介

项目选题为第四题，原题目要求如下：

> 制作完成图像合成软件，能够通过背景或背景建模等方法，对视频中人像进行完美分割，并与另外一个视频合并。（要求体现一定的新意，并综合使用多种图像处理技术。）

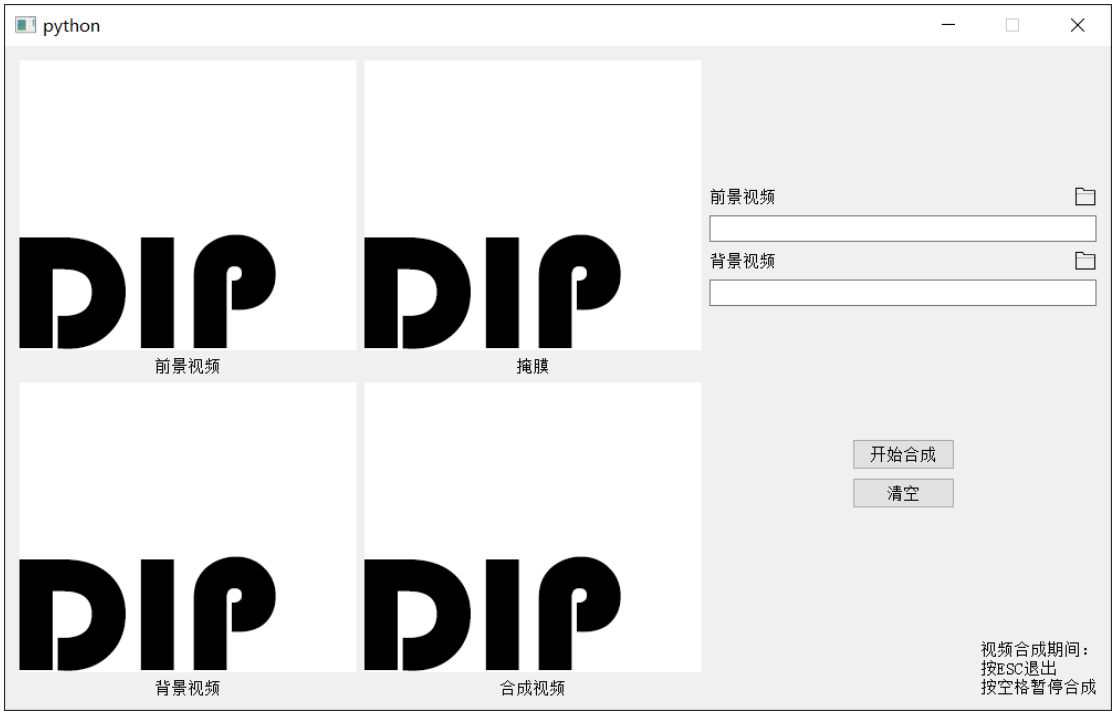
对于一段视频而言，如果镜头是固定的，那么视频的背景也会是固定不变的，因此视频中运动的部分可以被认为是前景，基于这一思想，我们可以通过运动与静止对视频实现前后景的分割。

首先我们要提取出视频中的静态区域，即视频的背景，这一过程就是背景建模，我们将所得到的背景模型和当前的图像进行比较，减去这些已知的背景就可以得到视频中运动的部分，即视频的前景。

抽取出视频的前景后，我们再想让它与另一视频合成就变得很容易了。本项目正是基于这一想法，运用经典的背景建模函数，实现原视频的前后景分离，进而实现两个视频的合成。并通过一些常用的数字图像处理方法提高前景提取的效果，同时利用 PyQt5 实现一个简单的交互界面以使软件更易于使用。

2 效果展示

（实际运行效果可见附录视频）


python
前景视频
掩膜
背景视频
合成视频
前景视频
背景视频
开始合成
清空
视频合成期间：
按ESC退出
按空格暂停合成


python
前景视频
掩膜
背景视频
合成视频
前景视频
D:/PythonCode/DIP_homework/vtest.avi
背景视频
onCode/DIP_homework/VID_20191227_170035.mp4
开始合成
清空
视频合成期间：
按ESC退出
按空格暂停合成

3 安装与使用说明

名称	日期	类型	大小	时长
.idea	2019/12/21 19:45	文件夹		
background.mp4	2019/12/27 17:00	MP4 文件	13,390 KB	00:00:18
folder.png	2019/12/22 20:04	PNG 图片文件	2 KB	
front.avi	2019/10/17 9:59	AVI 文件	7,942 KB	00:01:19
init.png	2019/12/24 17:54	PNG 图片文件	355 KB	
main.py	2019/12/27 22:15	JetBrains PyChar...	13 KB	

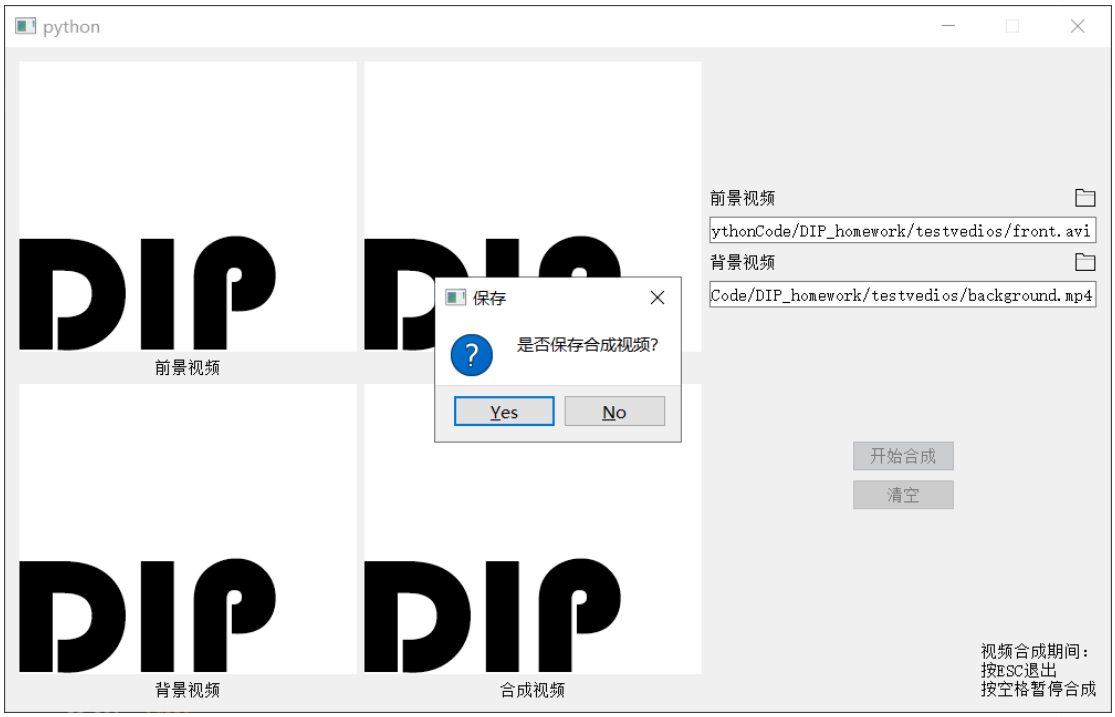
Pycharm 下打开附件中的项目 DIP_homework，运行 main.py 即可，两个图片文件（folder.png；init.png）是程序中会用到的两个图片资源，两个视频文件（background.mp4；front.avi）是供测试用的视频文件（为保证效果，建议将 front.avi 用作前景，将 background.mp4 用作背景）。

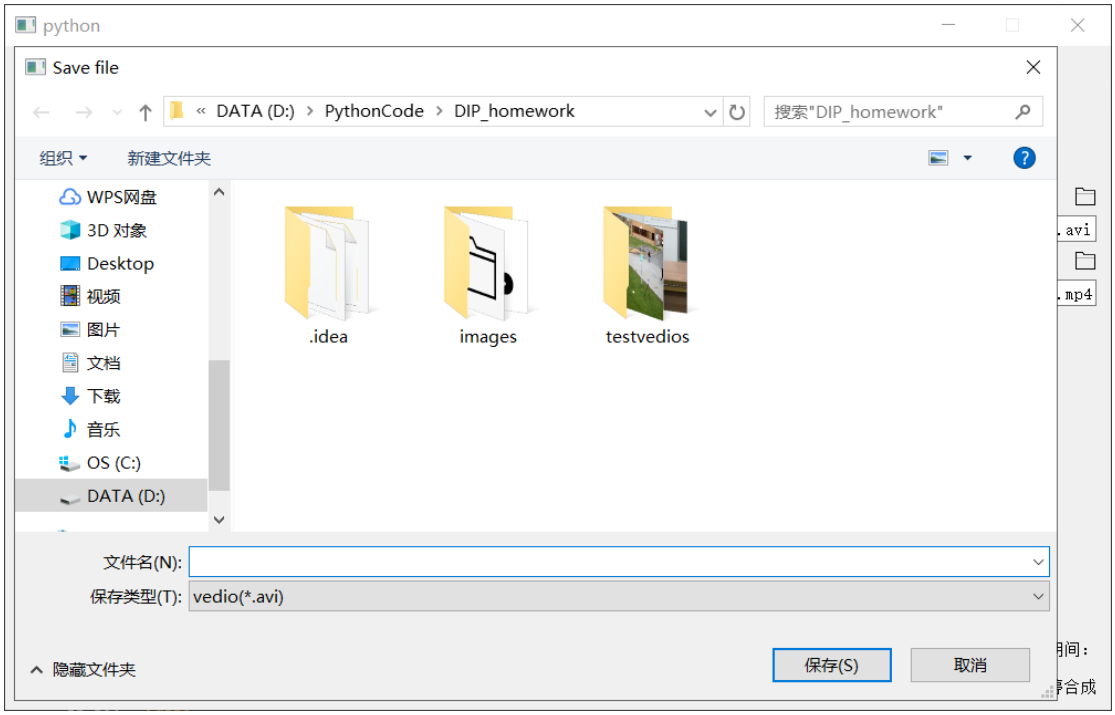
程序中使用了两个第三方库，cv2（图像处理）和 PyQt5（图形界面），运行前需确保已安装相关库。

使用说明：

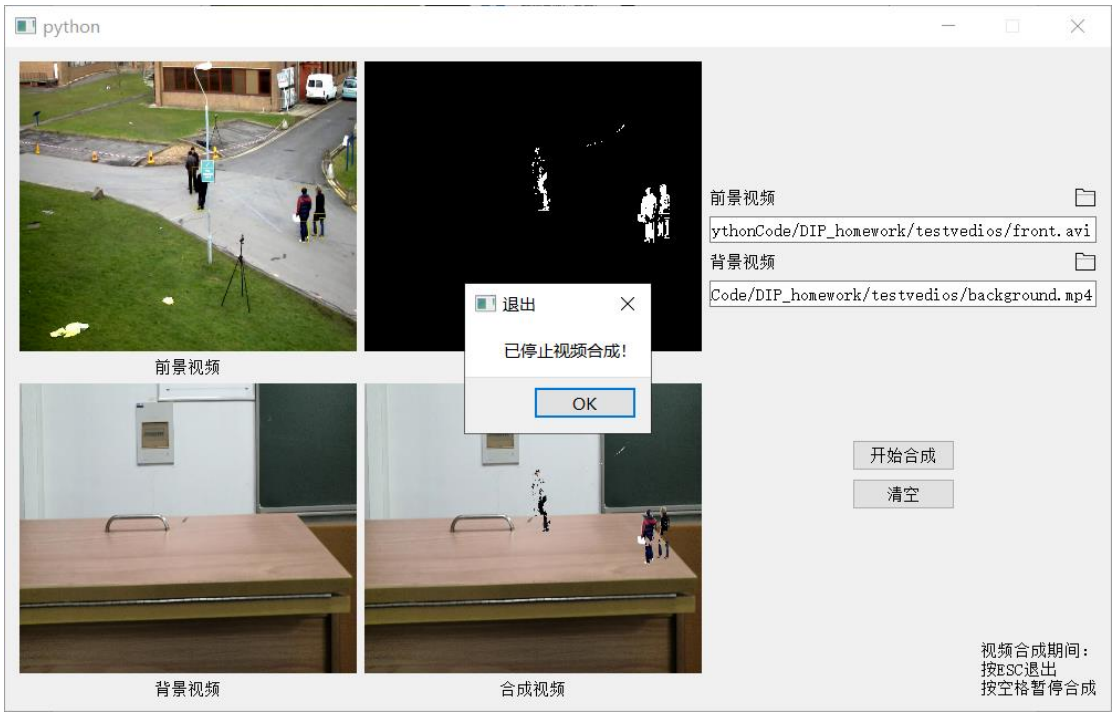
选择开始合成后（如未正确导入视频会提示报错），此时会弹出是否保存合成视频对话框，如选择保存则继续输入保存文件名（暂只支持.avi 格式），否则

直接开始合成视频。





视频开始合成后，可通过按空格键暂停合成（暂停后仍可继续），也可按 ESC 键退出合成。视频合成期间两个按钮会暂时失能，主要是防止与按键相冲突（当界面上有按纽时，按下空格键会被认为是点击按键）



4 设计文档

4.1 涉及知识点

4.1.1 混合高斯背景建模:

在提取运动目标之前必须先进行背景建模,背景建模的方法很多,如平均法、

最大值最小值统计法、单高斯建模法、加权平均法等，而混合高斯背景建模应该来说是比较成功的一种。关于混合高斯背景建模的原理如下：

图像中每个像素点的值（或特征）短时间内都是围绕在某一中心值一定距离内分布，通常，中心值可以用均值来代替，距离可以用方差来代替。这种分布是有规律的，根据统计定律，如果数据点足够多的话，是可以认为这些点呈正态分布，也就是高斯分布。根据这个特点，如果像素点的值偏离中心值较远，那么，这个像素点属于前景，如果像素点的值偏离中心值很近（在一定方差范围内），那么可以说这个点属于背景。理论上，如果不存在任何干扰的话，是可以准确区分前景和背景的。但是，现实往往不尽如人意，如果画面中光线变化的话，这个高斯分布的中心位置是会改变的。如果光线强度改变的话，在原来那个位置并没有无数个点供统计，因此，不符合大数定理，也就不能说那个点的分布满足正态分布了，只能说是近似为高斯分布。

混合高斯模型指这个像素点值存在多个中心位置，如来回摆动的树叶，波光粼粼的水面，闪烁的显示器，图像中特征边缘位置的抖动等，这些都会引起某个像素点会在多个中心位置聚集大量的点，每个位置便会产生一个高斯分布，四个以上的高斯分布其实并不常见，这便是混合高斯模型的由来。混合高斯背景建模主要用来解决背景像素点具有多峰特性的场合，如在智能交通场景中，解决视频画面抖动带来的干扰。

针对光线变化的问题，混合高斯模型通过比较当前像素点的值与高斯分布中心位置，选择一定的加权系数对当前高斯分布的中心位置进行更新，以便于适应缓慢的光线变化。

混合高斯背景建模方法优于其他背景建模方法的主要原因是：图像画面中，对于每个像素（也可以说是局部单元），其运动变化特性是不一样的，这才是混合高斯模型具有优势的主要原因。普通的二值化目标分割，整个画面采用同一阈值，无论这个阈值是固定的，还是自适应的，都存在本质性的缺陷，属于有缺陷的分割，这种算法带有不稳定性，无论怎么调整分割参数，也解决不了根本性的问题。因为，画面中每个部分，其实分割阈值是不一样的。一定得建立统计学的方法，进行局部分割，而混合高斯模型正是采用局部分割的一种算法。

4.1.2 腐蚀、膨胀和开运算：

腐蚀和膨胀运算是指在图像中移动一个结构元素，然后将结构元素与下面的二值图像在满足一定条件时对他们进行交、并等集合运算。腐蚀的作用是将图像的边缘腐蚀掉，消除目标边缘的毛刺；膨胀的作用是将图像的边缘扩大些，将目标的边缘或者是内部的坑填掉。

开运算是对图像先进行腐蚀运算再进行膨胀运算，使用开运算能够帮助我们除去图像上一些孤立的小点、毛刺和小桥，可以使目标表面更平滑，而总的位置和形状保持不变。

4.1.3 均值滤波和高斯滤波：

均值滤波是典型的线性滤波算法，它是指在图像上对目标像素给一个模板，该模板包括了其周围的临近像素（通常是一个 n*n 的方形框），再用模板中的全体像素的平均值来代替原来像素值。

均值滤波可以用来降低噪声，均值滤波器的主要应用是去除图像中的不相关细节，不相关是指与滤波器的模板相比较小的像素区域。模糊图片以便得到感兴趣物体的粗略描述，因此那些较小的物体的灰度就会与背景混合在一起，较大的物体则变的像斑点而易于检测。

高斯滤波就是对整幅图像进行加权平均的过程，每一个像素点的值，都由其本身和邻域内的其他像素值经过加权平均后得到。高斯滤波的具体操作是：用一个模板（或称卷积、掩模）扫描图像中的每一个像素，用模板确定的邻域内像素的加权平均灰度值去替代模板中心像素点的值。高斯滤波实质是对均值滤波的一个拓展，将均值滤波中简单地计算平均值改为计算他们的加权平均值，使结果更为合理。

4.2 主要技术

4.2.1 混合高斯背景建模

Opencv 封装了混合高斯背景建模函数：createBackgroundSubtractorMOG2()，混合高斯模型的学习方法是：

（1）首先初始化每个高斯模型矩阵参数；

（2）取视频中 T 帧数据图像用来训练高斯混合模型，来了第一个像素之后用它来当作第一个高斯分布；

（3）当后面再来新的像素值时，与前面已有的高斯分布的均值比较，如果该像素点的值与其模型均值差在 3 倍的方差内，则属于该分布，并对其进行参数更新；

（4）如果下一次来的像素不满足当前高斯分布，用它来创建一个新的高斯分布。

实际使用时代码如下：

```
fgbg = cv2.createBackgroundSubtractorMOG2()
fgmask = fgbg.apply(frame2)
```

第一行代码用于实例化一个混合高斯模型，第二行代码是对从视频中取出的图像使用上述模型，其返回的结果是一个二值图像，我们可以把它用作掩模以从原图像中获取我们所想要的部分。

4.2.2 开运算

Opencv 中也提供了封装好的对图像进行腐蚀膨胀组合操作的函数 morphologyEx，其具体使用方法如下：

```
element = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, element)
```

因为执行腐蚀和膨胀操作需要有一个模板，也就是我们常说的核，而函数 getStructuringElement()就是用来构造这样的结构化元素。它的第一个参数表示内核的形状，共有三种内核供选择，分别是矩形（MORPH_RECT）、交叉形（MORPH_CROSS）和椭圆形（MORPH_ELLIPSE），第二个参数表示核的大小，它还有一个可选参数，用以表示锚点的位置，其默认值是（-1，-1），表示锚点位于核的中心。因此本实验中采用的核的数组表示形式如下：

```
[[0,1,0]
 [1,1,1]
 [0,1,0]]
```

morphologyEx()函数用于对图像进行腐蚀和膨胀的组合操作，它常用的参数有三个：第一个是输入图像；第二个是操作的类型，它一共提供了八种操作类型，

这里我们选取 MORPH_OPEN，即开操作，对图像先进行腐蚀再进行膨胀操作；第三个参数是操作所用的结构元素，这里我们使用前面构造的内核 element。

4.2.3 二值化处理

上述混合高斯背景建模函数 createBackgroundSubtractorMOG2()较之第一代的一大改进就是实现了阴影的检测，但我们在实际使用中并不需要这一阴影，于是我们选择通过二值化的方法滤去人像的阴影，相关代码如下：

```
ret, fgmask = cv2.threshold(fgmask, 200, 255, cv2.THRESH_BINARY)
```

threshold()是 Opencv 封装的一个二值化处理函数，第二个参数是设定的阈值，第三个参数是当灰度值大于（或小于）阈值时将该灰度值赋成的值，第四个参数是规定的是当前二值化的方式，我们选择的是 THRESH_BINARY，表示大于阈值的部分会被置为 255，小于部分被置为 0。

4.2.4 轮廓检测

findContours()函数是 Opencv 中非常有用的轮廓检测和分割函数。在本实验中我们使用这一函数将我们识别出的前景中的物体标注出来，以实现一个简易的“追踪”效果，具体代码如下：

```
contours, hierarchy = cv2.findContours(fgmask.copy(),
                          cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
for cont in contours:
        Area = cv2.contourArea(cont)       # 计算轮廓面积
        if Area < 300:                     # 过滤面积较小的形状
            continue

        rect = cv2.boundingRect(cont)  # 提取矩形坐标

        cv2.rectangle(frame2, (rect[0], rect[1]), (rect[0]+rect[2],
                                        rect[1]+rect[3]), color, 1)
        cv2.rectangle(fgmask, (rect[0], rect[1]), (rect[0]+rect[2],
```

```
                                    rect[1]+rect[3]), (0xff, 0xff, 0xff), 1)
```

对于函数 findContours()，第一个返回值 contours 定义为“vector<vector<Point>> contours”，是一个双重向量（向量内每个元素保存了一组由连续的 Point 构成的点的集合的向量），每一组点集就是一个轮廓，有多少轮廓，contours 就有多少元素；第二个返回值 hierarchy 定义为“vector<Vec4i> hierarchy”，Vec4i 的定义：typedef Vec<int, 4> Vec4i;（向量内每个元素都包含了 4 个 int 型变量），所以从定义上看，hierarchy 是一个向量，向量内每个元素都是一个包含 4 个 int 型的数组。向量 hierarchy 内的元素和轮廓向量 contours 内的元素是一一对应的，向量的容量相同。hierarchy 内每个元素的 4 个 int 型变量是 hierarchy[i][0] ~ hierarchy[i][3]，分别表示当前轮廓 i 的后一个轮廓、前一个轮廓、父轮廓和内嵌轮廓的编号索引。如果当前轮廓没有对应的后一个轮廓、前一个轮廓、父轮廓和内嵌轮廓，则相应的 hierarchy[i][*]被置为-1（本实验中暂未使用这一参数）。

此外，我们使用了它的三个参数：

（1） 第一个参数就是输入的待检测图片，可以是灰度图，但更常用的是二值图像；

（2） 第二个参数定义轮廓的检索模式，取值如下：

CV_RETR_EXTERNAL：只检测最外围轮廓，包含在外围轮廓内的内围轮廓被忽略；

CV_RETR_LIST：检测所有的轮廓，包括内围、外围轮廓，但是检测到的轮廓不建立等级关系，彼此之间独立，没有等级关系，这就意味着这个检索模式下不存在父轮廓或内嵌轮廓，所以 hierarchy 向量内所有元素的第 3、第 4 个分量都会被置为-1；

CV_RETR_CCOMP: 检测所有的轮廓，但所有轮廓只建立两个等级关系，外围为顶层，若外围内的内围轮廓还包含了其他的轮廓信息，则内围内的所有轮廓均归属于顶层；

CV_RETR_TREE: 检测所有轮廓，所有轮廓建立一个等级树结构。外层轮廓包含内层轮廓，内层轮廓还可以继续包含内嵌轮廓。

（3） 第三个参数定义轮廓的近似方法，取值如下：

CV_CHAIN_APPROX_NONE：保存物体边界上所有连续的轮廓点到 contours 向量内；

CV_CHAIN_APPROX_SIMPLE：仅保存轮廓的拐点信息，把所有轮廓拐点处的点保存入 contours 向量内，拐点与拐点之间直线段上的信息点不予保留；

CV_CHAIN_APPROX_TC89_L1：使用 teh-Chinl chain 近似算法；

CV_CHAIN_APPROX_TC89_KCOS：使用 teh-Chinl chain 近似算法。

我们用 findContours()函数获取到所有轮廓后，对其进行遍历，忽略其中面积较小的区域（这些区域大概率是噪点，即使不是噪点，将其分割出来后也会因为较小而难以辨认）。boundingRect()函数是 Opencv 封装好的一个函数，它会用一个最小的矩形，把指定的包起来（读入的参数必须是 vector 或者 Mat 点集），然后我们再调用矩形绘制函数 rectangle()将其绘制到图像上。

至此，我们对原视频的一帧图像的主要处理工作就算完成了。下面，我们将介绍如何实现两个图像的合并。

4.2.5 利用掩模实现两幅图像的合并

先看核心代码：

```
fgmask = fgbg.apply(frame2)
fgmask_inv = cv2.bitwise_not(fgmask)


frame1_bg = cv2.bitwise_and(frame1, frame1, mask=fgmask_inv)
frame2_fg = cv2.bitwise_and(frame2, frame2, mask=fgmask)
dst = cv2.add(frame1_bg, frame2_fg)
```

数字图像处理中的掩模的概念是借鉴于 PCB 制版的过程，在半导体制造中，许多芯片工艺步骤采用光刻技术，用于这些步骤的图形“底片”称为掩膜（也称作“掩模”），其作用是：在硅片上选定的区域中对一个不透明的图形模板遮盖，继而下面的腐蚀或扩散将只影响选定的区域以外的区域。

类似的，如果我们在数字图像处理的过程中使用一幅二值图像作为掩模，那么为 0 的区域，即白色区域，就对应于盖板上的透明部分，为 1 的区域，即黑色区域，就对应于盖板上的不透明部分。

这里我们介绍 Opencv 中两个函数 bitwise_and()和 bitwise_not()，它们的作用分别是对两幅图片进行按位与和对一幅图片按位取非。我们用

bitwise_and()函数对两幅相同的图片（原视频一帧图像）进行操作，显然结果还是原图片，但如果我们加入了我们之前得到的掩模 fgmask，我们得到的就只有我们感兴趣的前景区域；我们对掩膜使用一次 bitwise_not()函数，那么它的黑白区域将互换，我们将这一新得到的掩模用于背景视频，就可以获取我们想要的新背景。前后两次得到的图片应该是完美互补的，我们把他们相加（使用 add()函数），就可以得到我们想要的合成图片，即代码中的 dst。

4.2.6 视频的保存

Opencv 提供了保存视频用的函数，相关代码抽取如下：

```
fourcc = cv2.VideoWriter_fourcc(*'XVID')  # 设置保存图片格式
out = cv2.VideoWriter(save_address, fourcc, 10.0, (cols, rows), True)
out.write(dst)
out.release()
```

代码比较简单，易于理解，这里我们不做过多解释。

4.2.7 使用 PyQt5 设计一个简单的图形界面

为了使软件便于使用，我们决定开发一个简单的交互式图形界面，PyQt5 是 Python 中常用的界面开发工具，其使用和理解也类似于大多数界面开发工具。

本实验中我们主要使用了 PyQt5 的常见组件：QLabel, QVBoxLayout, QGridLayout, QLineEdit, QHBoxLayout, QPushButton, QFileDialog, QMessageBox 等，通过构造一个界面类，以及在主函数中调用这一界面类实现界面的初始化。因为不是本实验的重点内容，所以我们不做详细介绍。

在开发界面的过程中遇到一个小问题，就是我们要在 QLabel 中展示视频，但是 QLabel 支持的图片格式是 QPixmap，而我们图像处理过程中使用的图片格式是 cv2，因此需要对图片格式进行转换，下面是所用的转换函数：

```
def convert_image(image):
    image = cv2.resize(image, (305, 285))


    if len(image.shape) == 2:
        ret, image = cv2.threshold(image, 0, 255, cv2.THRESH_OTSU | cv2.THRESH_BINARY)
        image = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)
```

```
    cv2.cvtColor(image, cv2.COLOR_BGR2RGB, image)
    qt_image = QImage(image.data, image.shape[1], image.shape[0], image.shape[1]*3,
QImage.Format_RGB888)
    return qt_image
```

5 特色与创新

（1） 搭配图像界面，交互良好；

（2） 对前景进行标记，实现简单的对象跟踪；

（3） 实现对视频的保存；

（4） 使用多种图像降噪方法以提高视频处理效果；

6 后期改进想法

软件目前的处理效果稳定性较差，即处理的效果受视频本身影响较大，这一点单纯依靠图像处理方法并不能很好的解决，因此我们觉得应该把降噪、优化的工作交给用户来做，即我们设置好相关方法，但方法的选择和参数的设置由用户根据实际情况来决定。