**实验一：混合图像**

# 实验环境

python：3.7.4

numpy：1.16.4

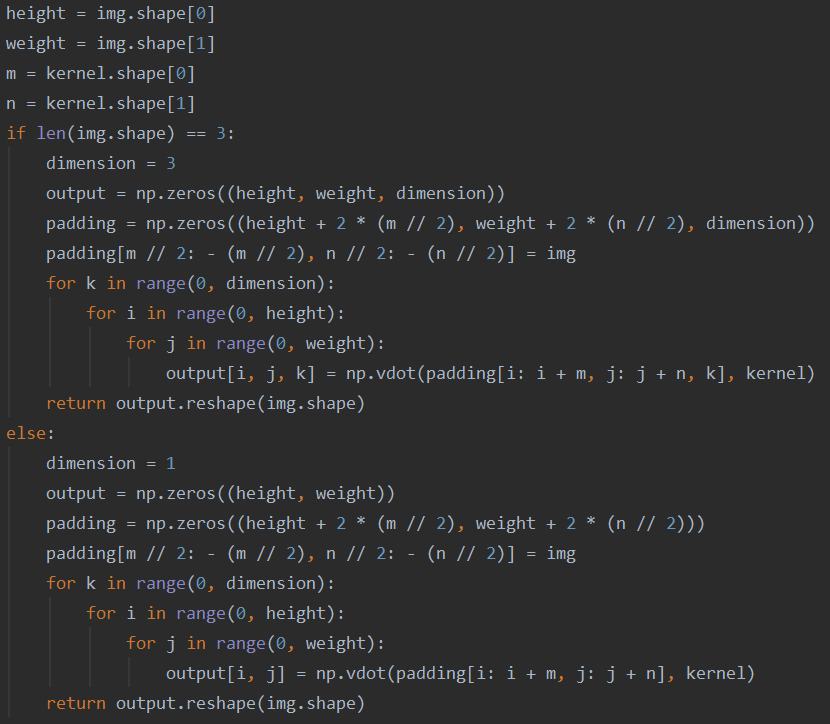
opencv：3.4.2

pillow：6.1.0

pytest：5.0.1

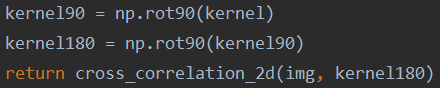
# 实现的函数

## 互相关函数



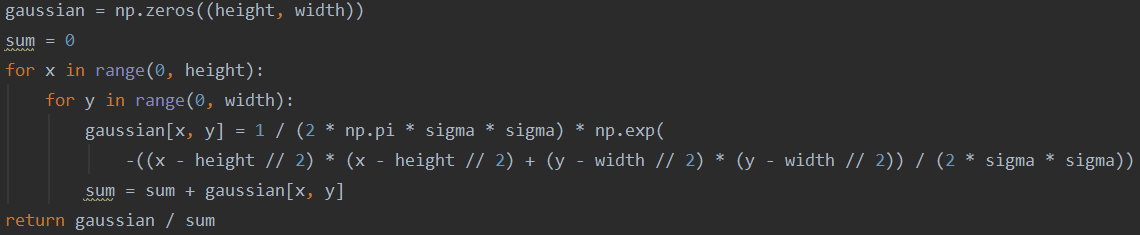
* 首先通过len(img.shape)判断图像是灰度图像还是RGB图像。
* 通过计算kernel的大小得出填充区域的大小。创建一个填充之后大小的矩阵并将img中的数值赋值到相应位置。
* 通过三重循环实现kernel与对应位置相乘并求和。

## 卷积函数



* 通过调用numpy.rot90()实现kernel矩阵旋转90度。
* 再次调用实现kernel矩阵旋转180度。

## 高斯核函数



* 通过二维高斯函数公式，生成高斯函数并进行归一化处理。

## 低通滤波函数



* 将img矩阵与高斯核进行卷积即可得到低通滤波后的图像。

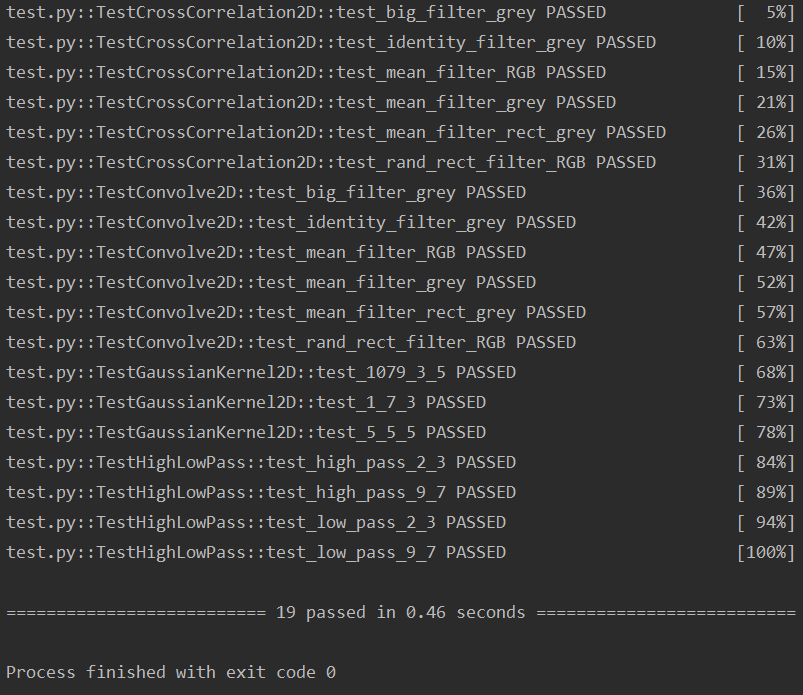
## 高通滤波函数



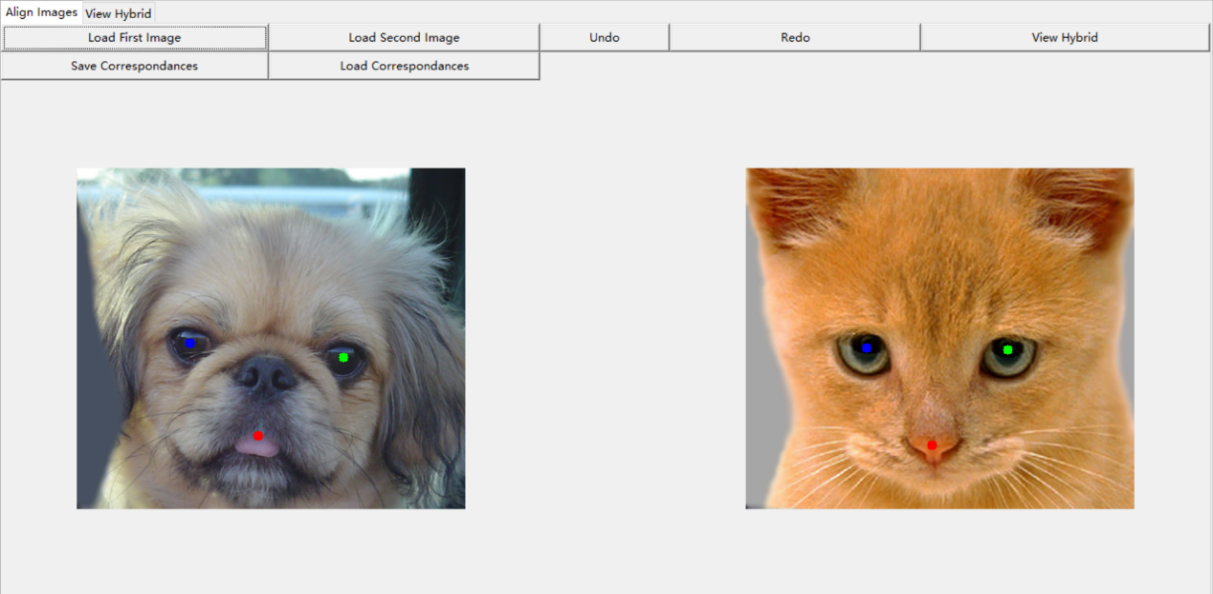
* 将原图像减去低通滤波后的图像即可得到高通滤波后的图像。

# 实验结果

## test.py测试结果

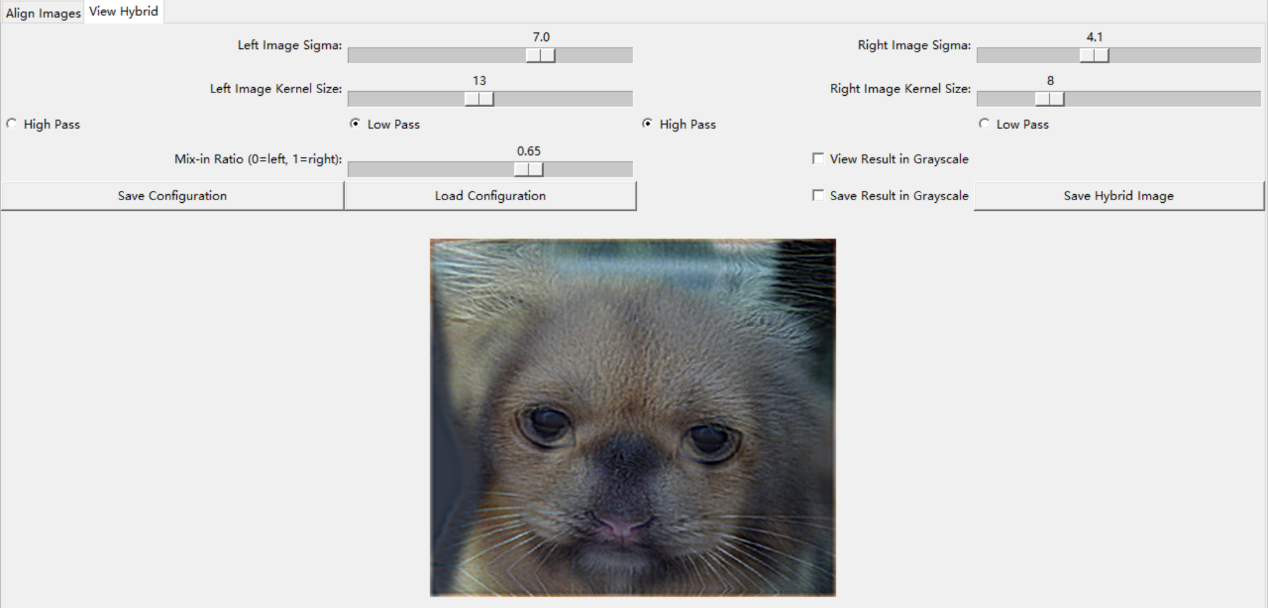


## 原始图像



* 左边图像为低通，右边图像为高通。

## 参数设置



* 所有参数均为resources\sample-config.json中所设置的参数值。
* 个人测试各个参数在resources\sample-config.json中所设置的参数值附近所得出的图像最佳，故采用resources\sample-config.json中所设置的参数值。

## 混合后图像



* 不难看出，混合后的图像放大之后还是可以看出猫的样子，而缩小之后基本看不出猫的样子。

# 实验心得

在第一次实现互相关函数的时候采用了如下图的五个循环的方法，在调参的时候发现不论是调整sigma还是调整kernel size都需要将近一分钟的时间，整个混合过程需要五六分钟。在上机实验的时候与学长交流之后改为三个循环，处理效率大大增加，每个参数的调整所需时间降低为十秒以下。大大提高了整个混合的效率。在之前写程序的时候因为程序规模较小，我往往不考虑算法的效率问题，但通过这次试验，我第一次体会到一个好的算法的重要性。

