电 子 科 技 大 学

实　验　报　告

学生姓名：刘文晨　　学 号：2018080901006　　指导教师：沈复民，徐行

一、实验项目名称：

图像过滤和混合图像（Image Filtering and Hybrid Images）

二、实验原理：

1. 图像的频率、高频与低频

图像可以视为一个定义为二维平面上的信号，该信号的幅值对应于像素的灰度值（对于彩色图像则是 RGB 三个分量）。图像的频率是灰度值变化剧烈程度的指标，是灰度在平面空间上的梯度，它反映了图像的像素灰度在空间中变化的情况。

对图像而言，低频分量代表着图像中亮度或者灰度值变化缓慢的区域，也就是图像中大片平坦的区域，描述了图像的主要部分，是对整幅图像强度的综合度量。高频分量对应着图像变化剧烈的部分，也就是图像的边缘（轮廓）或者噪声以及细节部分。

2. 图像过滤

图像过滤，即在尽量保留图像细节特征的条件下对目标图像的噪声进行抑制，是图像预处理中不可缺少的操作，其处理效果的好坏将直接影响到后续图像处理和分析的有效性和可靠性。

3. 滤波器

滤波器可分为高通滤波器与低通滤波器。高通滤波器可以检测图像中尖锐、变化明显的地方；低通滤波器可以让图像变得光滑，滤除图像中的噪声。常见的滤波器有：非线性滤波器、中值滤波、形态学滤波器和双边滤波。

4. 图像融合

图像融合是指将多源信道所采集到的关于同一目标的图像数据经过图像处理和计算机技术等，最大限度的提取各自信道中的有利信息，最后综合成高质量的图像，以提高图像信息的利用率、改善计算机解译精度和可靠性、提升原始图像的空间分辨率和光谱分辨率，利于监测。

三、实验目的：

对所给图像分别进行高通和低通滤波，并通过图像融合得到混合图像。

四、实验内容：

1. 了解并学习图像过滤与图像融合的相关知识与操作。
2. 图像过滤：完善相关代码，对所给图像进行滤波处理，分别得到图像的高频信号与低频信号。
3. 图像融合：融合图像的高频信号与低频信号，输出融合后的图像。

五、实验步骤：

1. 图像过滤

完善 my_imfilter()函数：

```
1.  def my_imfilter(image, filter):
2.      """
3.      Your function should meet the requirements laid out on the project webpa
    ge.
4.      Apply a filter to an image. Return the filtered image.
5.      Inputs:
6.      - image -> numpy nd-array of dim (m, n, c)
7.      - filter -> numpy nd-array of odd dim (k, l)
8.      Returns
9.      - filtered_image -> numpy nd-array of dim (m, n, c)
10.     Errors if:
11.     - filter has any even dimension -> raise an Exception with a suitable er
    ror message.
12.     """
13.     filtered_image = np.asarray([0])
14.
15.     pad_x = (filter.shape[0] - 1) // 2
```

```
    pad_y = (filter.shape[1] - 1) // 2

    image_pad = np.pad(image, (
        (pad_x, pad_x),
        (pad_y, pad_y),
        (0, 0)), 'constant')

    filtered_image_height = image_pad.shape[0] - filter.shape[0] + 1
    filtered_image_width = image_pad.shape[1] - filter.shape[1] + 1
    filtered_image = np.zeros([filtered_image_height, filtered_image_width, image.shape[2]])

    for k in range(image.shape[2]):
        for i in range(filtered_image_height):
            for j in range(filtered_image_width):
                filtered_image[i, j, k] = np.sum(
                    np.multiply(image_pad[i:i + filter.shape[0], j:j + filter.shape[1], k], filter))

    return filtered_image
```

2. 图像融合

完善 gen_hybrid_image()函数：

```
def gen_hybrid_image(image1, image2, cutoff_frequency):
    """
     Inputs:
     - image1 -> The image from which to take the low frequencies.
     - image2 -> The image from which to take the high frequencies.
     - cutoff_frequency -> The standard deviation, in pixels, of the Gaussian
                           blur that will remove high frequencies.

     Task:
     - Use my_imfilter to create 'low_frequencies' and 'high_frequencies'.
     - Combine them to create 'hybrid_image'.
    """

    assert image1.shape[0] == image2.shape[0]
    assert image1.shape[1] == image2.shape[1]
    assert image1.shape[2] == image2.shape[2]

    # Steps:
```

```
    # (1) Remove the high frequencies from image1 by blurring it. The amount of
    #     blur that works best will vary with different image pairs
    # generate a 1x(2k+1) gaussian kernel with mean=0 and sigma = s, see https://stackoverflow.com/questions/17190649/how-to-obtain-a-gaussian-filter-in-python
    s, k = cutoff_frequency, cutoff_frequency * 2
    probs = np.asarray([exp(-z * z / (2 * s * s)) / sqrt(2 * pi * s * s) for z in range(-k, k + 1)], dtype=np.float32)
    kernel = np.outer(probs, probs)

    # Your code here:
    low_frequencies = my_imfilter(image1, kernel)  # Replace with your implementation

    # (2) Remove the low frequencies from image2. The easiest way to do this is to
    #     subtract a blurred version of image2 from the original version of image2.
    #     This will give you an image centered at zero with negative values.

    # Your code here #
    high_frequencies = image2 - my_imfilter(image2, kernel)  # Replace with your implementation

    # (3) Combine the high frequencies and low frequencies
    # Your code here #
    hybrid_image = low_frequencies + high_frequencies  # Replace with your implementation

    # (4) At this point, you need to be aware that values larger than 1.0
    # or less than 0.0 may cause issues in the functions in Python for saving
    # images to disk. These are called in proj1_part2 after the call to
    # gen_hybrid_image().
    # One option is to clip (also called clamp) all values below 0.0 to 0.0,

    # and all values larger than 1.0 to 1.0.

    return low_frequencies, high_frequencies, hybrid_image
```

六、实验数据及结果分析：

以猫和狗图像为例，两个原始图像如图 1 所示，其余实验结果见于附件 1。

图 1 原始图像

上述图像经过高通和低通滤波处理后的图像如下所示：

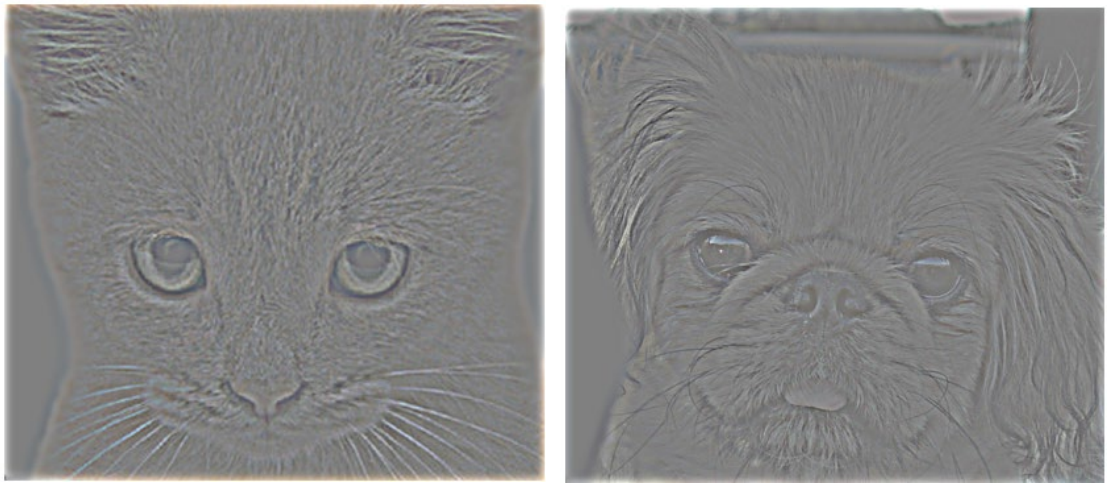

图 2 高通滤波图像

图 3 低通滤波图像

提取猫图像的高频信号，再提取狗图像的低频信号，进行图像融合，混合图像如图 4 所示。

图 4 混合图像

图 5 不同大小的混合图像

七、实验结论：

高通滤波可以检测图像中尖锐、变化明显的地方；低通滤波可以让图像变得光滑，滤除图像中的噪声，使图像变得模糊。

从最后融合的图像可以看出，当图像较大时，即近看，看到的是猫。这是因为融合图像的高频来自猫。当图像逐渐变小时，即变成远看，看到的是狗，这是因为融合图像的低频信息来自狗，低频信息代表远看时的信息，高频信息代表近看时的信息。其他四对生成的四个混合图像也证实了这一理论。

八、总结及心得体会：

通过完善相关代码实现了对所给图像分别进行高通和低通滤波，并通过图像融合得到混合图像，得到了低频信息代表远看时的信息，高频信息代表近看时的信息这一结论。

九、对本实验过程及方法的改进建议：

提供实验指导书。

报告评分：

指导教师签字：

附件 1


P.G.C


P.G.C