电 子 科 技 大 学

实 验 报 告

学生姓名：郭宇航　　学 号：2016100104014　　指导教师：徐行

一、实验项目名称：高低通滤波图像融合

二、实验原理：

频域滤波：

频域滤波是指在频率域对图像进行处理的一种方法，主要步骤如下：

图 1：频域滤波流程图

滤波后的结果显示低通滤波可以过滤高频信息（细节信息），留下的低频信息表示图像的概况。而高通滤波相反，起到的是锐化的作用。下面一一介绍：

（1）低通（平滑）滤波器：

理想中的低通滤波器的模板为：

$$H(u,v)=\begin{cases}1, & D(u,v)\le D_0\\ 0, & D(u,v)>D_0\end{cases}$$

其中，D_0 表示通带半径，$D(u,v)$是频谱中心的距离，计算公式如下：

$$D(u,v)=\sqrt{(u-M/2)^2+(v-N/2)^2}$$

其中 M 和 N 表示频谱图像的大小，$(M/2,N/2)$即为频谱中心.

（2）高通（锐化）滤波器：

正像低通滤波使得图像变得模糊那样，相反的处理过程——高通滤波，通过削弱傅里叶变换的低频以及高频相对不变来锐化图像，假设给定低频滤波器的传递函数为 $H_{LP}(u,v)$，则对应的高频滤波器表示为：

$$H_{HP}(u,v)=1-H_{LP}(u,v)$$

本次实验中我们使用到的是 Gaussian 滤波器，Gaussian 低通滤波函数表示为：

$$H(u,v)=e^{-\frac{D^2(u,v)}{2D_0^2}}$$

高斯滤波具有很多重要的性质：

（1） 旋转对称性：滤波器在各个方向上的平滑效果是相同的。

（2） 单值函数：高斯滤波器用像素邻域的加权均值来代替该点的像素值，而每

一邻域像素点权值是随该点与中心点的距离单调增减的。

（3） 傅立叶变换频谱为单瓣：高斯函数付立叶变换的单瓣意味着平滑图像不会被不需要的高频信号所污染，同时保留了大部分所需信号。

（4） 宽度由 σ 参数表征，σ 决定了平滑程度，高斯滤波器的频带就越宽，平滑程度就越好．通过调节平滑程度参数 σ，可在图像特征过分模糊(过平滑)与平滑图像中由于噪声和细纹理所引起的过多的不希望突变量(欠平滑)之间取得折衷。

（5） 可分离性：二维高斯函数卷积可以分两步来进行，首先将图像与一维高斯函数进行卷积，然后将卷积结果与方向垂直的相同一维高斯函数卷积。计算复杂度线性增长。

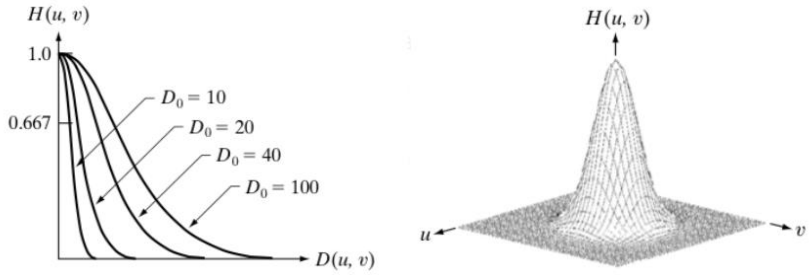

图 2：高斯滤波几何直观

三、实验目的：

对不同的图像分别进行高低通滤波，然后融合图片并进行下采样缩放给出效果示意图。

四、实验内容：

（1）了解图像处理领域的滤波操作的过程。

（2）完成代码部分的填充，主要包括 my_imfilter()以及 gen_hybrid_image()函数的完善。

（3）给出融合的图片的结果。

五、实验步骤：

一、完善 my_imfilter()函数：

my_imfilter()函数主要给出的是在已知滤波器的情况下，如何对已有的 image 进行卷积得到滤波后的图像。下面给出这一部分的代码：

```
def my_imfilter(image, filter):
  """
  Your function should meet the requirements laid out on the project webpage.
  Apply a filter to an image. Return the filtered image.
  Inputs:
  - image -> numpy nd-array of dim (m, n, c)
```

```
  - filter -> numpy nd-array of odd dim (k, l)
  Returns
  - filtered_image -> numpy nd-array of dim (m, n, c)
  Errors if:
  - filter has any even dimension -> raise an Exception with a suitable error
message.
  """
  #print('原图的尺寸', image.shape)
  #print('卷积核的尺寸', filter.shape)
  '''
  :param image:image 是一个三维的矩阵：m*n*c，其中 m*n 是单个通道的尺寸，c 是通
道的数量
  :param filter:卷积核的尺寸 k*l
  :return:返回一个卷积后的 feature map
  '''
  # 这里处理的卷积都是输出原尺寸，即先对原图进行 pad 操作，然后再卷积
  # 首先进行 padding
  image_padding = np.pad(image, (
  ((filter.shape[0] - 1) // 2, (filter.shape[0] - 1) // 2), ((filter.shape[1]
- 1) // 2, (filter.shape[1] - 1) // 2),
  (0, 0)), 'constant')
  #print('padding 之后图像的尺寸', image_padding.shape)
  # 计算输出图像的尺寸
  output_image_height = image_padding.shape[0] - filter.shape[0] + 1
  output_image_width = image_padding.shape[1] - filter.shape[1] + 1
  output_image = np.zeros([output_image_height, output_image_width,
image.shape[2]])
  time_start = time.clock()
  for k in range(image.shape[2]):
    for i in range(output_image_height):
      for j in range(output_image_width):
        output_image[i, j, k] = np.sum(
          np.multiply(image_padding[i:i + filter.shape[0], j:j +
filter.shape[1], k], filter))
  time_finish = time.clock()
  #print('卷积用时为：', time_finish - time_start)
  #print('卷积之后的图像尺寸', output_image.shape)
  return output_image
```

由于代码的实现要求是图像的尺寸：m*n*c，其中 m*n 是单通道的尺寸，c 是色彩通道数，滤波器的尺寸是 k*l，要求输出的尺寸仍然为 m*n*c，所以首先需要对原始图像周围进行补零的操作，补零的个数根据不添零时的输出尺寸反推，如果不添加零，则输出的尺寸为：(m-k+1)*(n-l+1)*c，不考虑 stride>1 的情况。因此补零的个数就是 filter 的长宽分别-1 再对 2 下取整的结果。

补零之后直接使用 for 循环进行卷积操作，对应区域对应的值相乘再相加，最终得到了滤波卷积之后的图像。

二、完善 get_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'.
  """
  #断言：判断两个输入进来的图片的尺寸大小是否一样
  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-f
ilter-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)
  #print('卷积核的显示',kernel)
  #print('卷积核的尺寸大小：',kernel.shape)
  # Your code here:
  low_frequencies_1 = cv2.filter2D(src=image1,ddepth=-1,kernel=kernel)
  #print("cv2 的低通滤波器得到的图像尺寸",low_frequencies_1.shape)
  low_frequencies = my_imfilter(image1,kernel)
  low_frequencies = low_frequencies.astype(np.uint8)
  #print("低通滤波器得到的图像尺寸",low_frequencies.shape)
  cv2.imshow('low_frequencies',low_frequencies)

  # (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 -
cv2.filter2D(src=image2,ddepth=-1,kernel=kernel)
  high_frequencies = image2 - my_imfilter(image2,kernel)
  high_frequencies = high_frequencies.astype(np.uint8)
  #print("高通滤波器得到的图像尺寸",high_frequencies.shape)
  cv2.imshow('high_frequencies',high_frequencies)
  # 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.
  for k in range(hybrid_image.shape[0]):
    for i in range (hybrid_image.shape[1]):
      for j in range (hybrid_image.shape[2]):
        hybrid_image[k,i,j] = clamp(hybrid_image[k,i,j])
  cv2.imshow('result',hybrid_image)
  cv2.waitKey(0)
  cv2.destroyAllWindows()
  return low_frequencies, high_frequencies, hybrid_image
```

这一部分的代码主要需要补全的是基于已经构造好的滤波器(kernel)进行高通滤波和低通滤波，prob 构造了一个高斯分布，并基于此高斯分布构造了低通滤波器，这里使用了两种方法得到了滤波的结果，第一种是 cv2 库函数中的 filter2D 函数得到滤波图像，另外一种是使用自己刚刚写好的 my_infilter()函数得到的滤波图像。

在得到滤波图像之后还需要检查图像的像素点的值是否越界了，如果越界了则需要修改该像素点的值，这里主要通过一个简单的 clamp()函数构造出来,代码如下：

```
def clamp(pixel_value):
    if pixel_value >= 255:
        return 255
    if pixel_value <= 0:
        return 0
    else:
        return pixel_value
```

处理过后我们可以使用 cv2 库中的 imshow()函数查看滤波之后的结果。

对于需要进行高通滤波的图像来说，做法比较简单，首先使用低通滤波对该图像进行卷积，得到低通滤波的图像后使用原图像像素点矩阵对低通滤波后的图像像素点矩阵进行对应位置的减法运算得到图像的高通滤波的结果。

在代码中的体现就是：

high_frequencies = image2 - my_imfilter(image2,kernel)

high_frequencies = high_frequencies.astype(np.uint8)

最终将不同图像的高低通滤波进行融合，即进行相加求和的操作，得到 hybrid 的图像。

六、实验数据及结果分析：

原始需要融合的图像如下：

图 3：实验原始图像

左图是一只猫，我们需要对它进行高通滤波的处理，即锐化细节处理。

右图是一只狗，我们需要对它进行低通滤波，即模糊平滑处理。

下面给出滤波得到的结果：

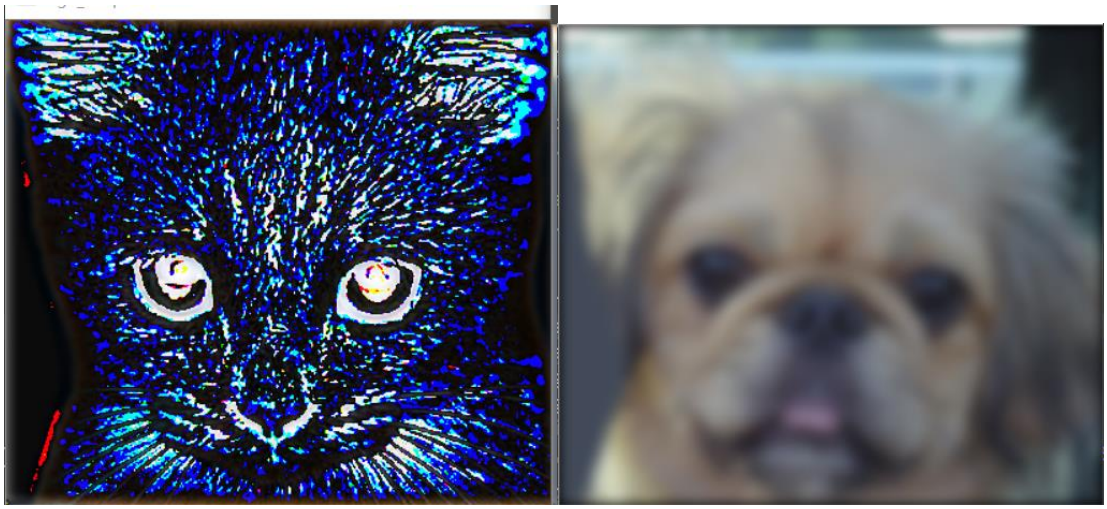

图 4：高低通滤波得到的图像

上图给出了猫的图片的高通滤波和狗的图片的低通滤波的结果。我们看到高通滤波确实起到了锐化一些纹理细节的作用，而低通滤波则是将图像模糊化处理

了。

除此之外，pro1_part1.py 和 pro1_part2.py 还给出了一些其他的滤波器得到的高低通滤波的结果。下面以猫的图像为例给出一些示例：

Sobel 和 laplacian 得到的高通滤波的结果：

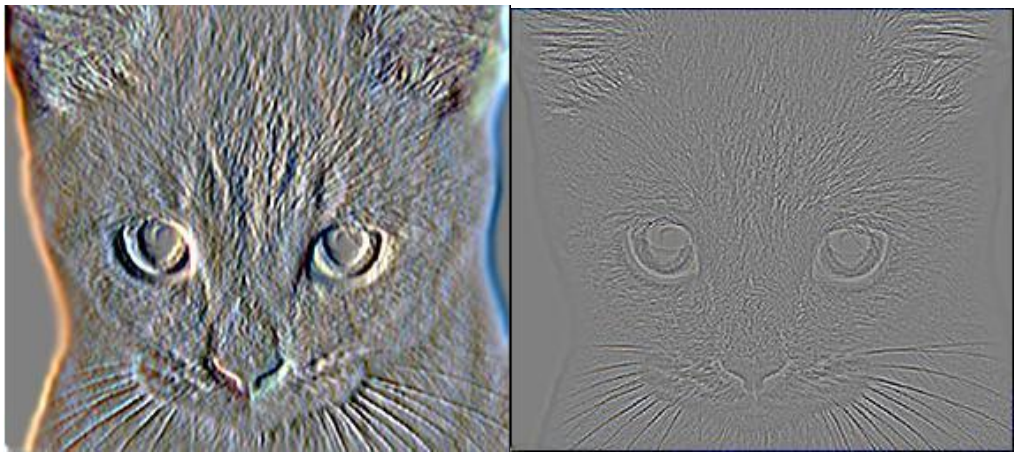

图 5：sobel 和 laplacian 高通滤波得到的图像

最终给出高低通滤波得到的融合的图像以及对图像进行缩放呈现出来的效果：

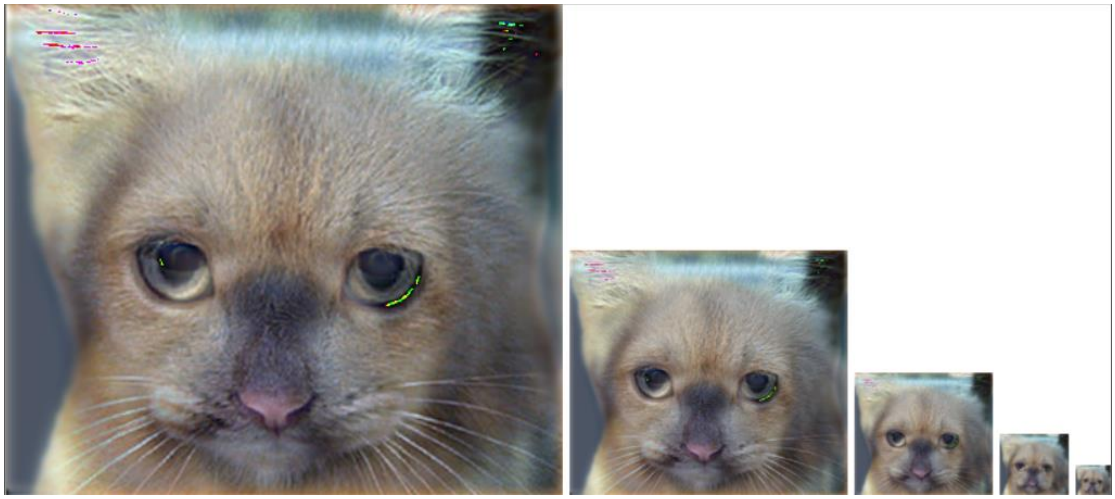

图 6：Hybrid 融合之后的图像

比较神奇的现象就是，融合之后的图像，在图像尺寸比较大的时候呈现出的结果更加趋向于猫的形状，而当图像的尺寸缩放 2,4,8,16 倍的时候，越来越体现出狗的图片的特征。

七、实验结论：

分析实验的结果，首先看滤波器是如何在图像上工作的，图像某种程度上也可以表示为一种波，所以是可以通过波的算法来处理图像的。对于一张 400*400 的图像来说，取一个 1*400 的行向量，我们可以根据像素点的值绘制图像：

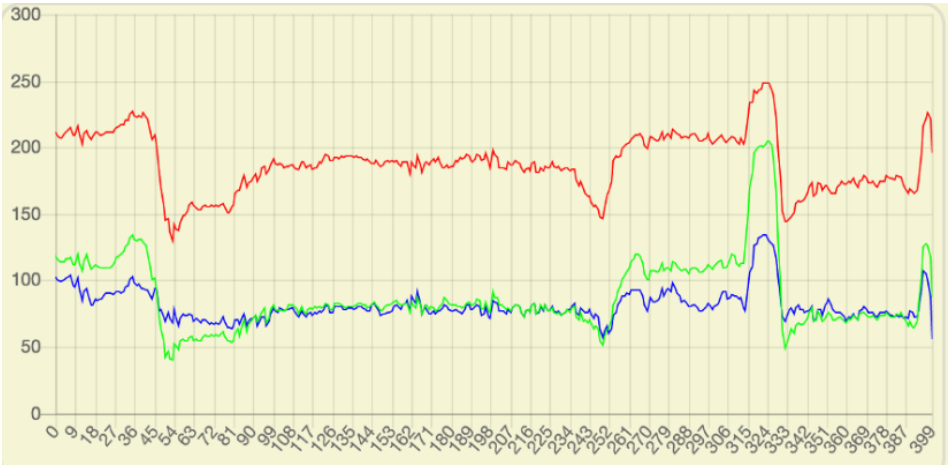

图 7：图像的波的表示

因此我们是可以在频域上对图像进行处理的。

对于低通滤波：

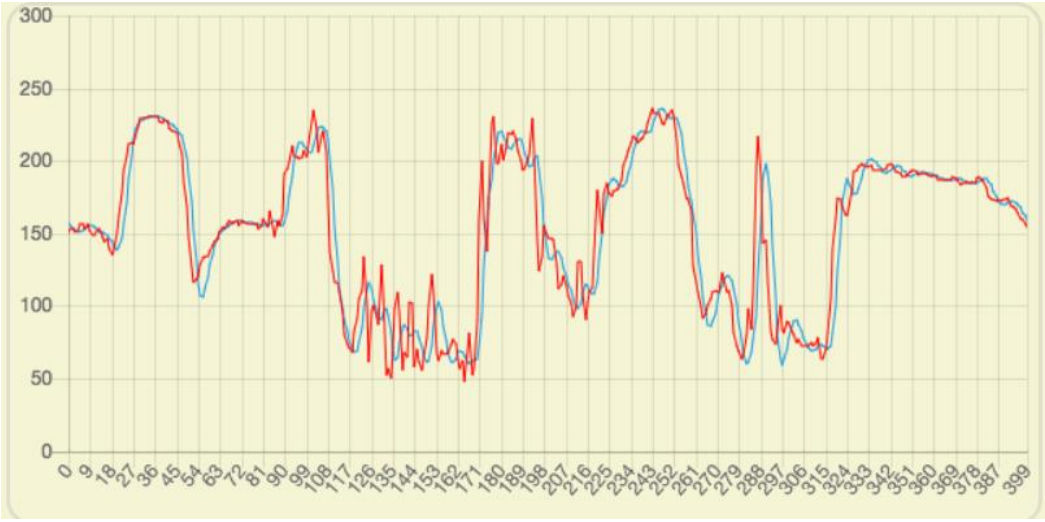

图 8：低通滤波后图像波形

对于高通滤波：

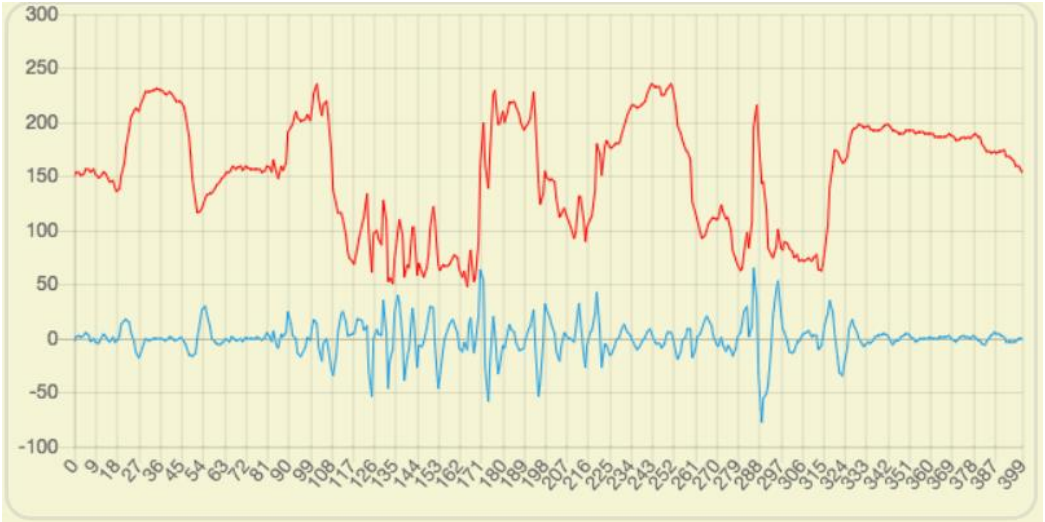

图 9：高通滤波后图像波形

观察上述的图 8 和图 9，其中红线为原始的图像，而蓝线表示的是滤波后的图像，首先看低通滤波，我们不难发现通过低通滤波后的图像波呈现的是更加平滑的曲线，减弱了图像中的很多像素点变化较大的位置；高通滤波则是相反的，得到的图像波的结果是将所有图像中的像素点的波动全部保留下来，即那些变化最快速的最剧烈的区域。

关于图像融合的理解，在融合后的图像中，我们发现当图像的尺寸越大，高通滤波器得到的图像在其中显现的更加明显，个人觉得是因为高通滤波留下的是一些细节的纹理信息，线条信息，因此可以看上去更加的清楚。而当图像尺寸越来越小的时候，人眼只会观察到最基本的图像的大致情况，因此低通滤波的对图像大致轮廓的描绘会更加凸显。

八、总结及心得体会：

通过实验真正的理解了图像的滤波操作，学会了一些图像处理的基本的操作。在实验中也踩了不少的坑，比如说使用 cv2 中的 imshow()函数时，存在两种不同的情况，如果图像的像素点矩阵所有元素均为 int 型，则图像得到的结果是按照 0-255 的模式显示图像，而如果是浮点数则自动默认为将图像归一化后的结果，范围就在 0-1 之间。之前在实验中将两种模式混用导致图像不能正确的显示，后来才明白其中的道理。

九、对本实验过程及方法的改进建议：

1.可以使用不同的滤波器和图像对比之间的差别。

报告评分：

指导教师签字：