Hybrid Images

Introduction

Hybrid images was proposed in the SIGGRAPH 2006 paper (https://www.researchgate.net/publication/220184425_Hybrid_images) by Oliva, Torralba, and Schyns. Here (http://olivalab.mit.edu/hybridimage.htm) is the webpage of their work.

Hybrid images are static images that change in interpretation as a function of the viewing distance. The basic idea is that high frequency tends to dominate perception when it is available, but, at a distance, only the low frequency (smooth) part of the signal can be seen. By blending the high frequency portion of one image with the low-frequency portion of another, you get a hybrid image that leads to different interpretations at different distances.

Low-frequency parts of the image can be understood as "contours", such as the shape of the face. "high attached microdermabrasion" 微晶磨皮

The high-frequency parts of the image can be understood as "details", such as wrinkles and spots on the face.

Therefore, we often say that the low-frequency part of the image is obtained after the image is blurred, and the image is sharpened to make the high-frequency information of the image more.

Basic steps

The overall idea of the implementation is simple-superimpose a picture with only low-frequency information and an image with only high-frequency information. Specific steps are as follows:

1. Prepare low-frequency filters (usually Gaussian blur filter).
2. Convolution of each dimension of the first image with Gaussian filter.
3. Convolution of each dimension of the second image with Gaussian filter, and subtract the filtered image from the original image to obtain the high-frequency image. ! NOTE that the Gaussian filter in this step may have different standard deviation from the above one. You need to try different values to produce satisfatory results.
4. Add the two processed images to get a hybrid image.

Implementation details

1. Read a pair of images and alignment

Read a pair of images. You can use the the given images for test, but for submission, you have to use your own images. The image pair should be well aligned for satisfatory results. You need to mannualy select two points on each image and calculate the rotation and scaling parameters.

For example, the above cat and dog are not aligned. We need to rotate and scale the dog image to align with the cat. We select the centers of the two eyes on the dog image, then we can caculate the rotation angle and the scaling factor.

This method exploys two points on each image. You need to think of appropriate steps to do the alignment. The following steps are just one possible way:

1. Select two points A1 and A2 on image A for alignement; and selcet two points B1 and B2 on image B;
2. Translate image B so that B1 is aligned with A1;
3. Rotate image B to make the line B1B2 have the same direction with line A1A2;
4. Scale image B, so that the distance between B1 and B2 is equal to the distance between A1 and A2;
5. Crop the images to make the two image have the same size.

Please write your code below to align your image pair. Note that, you can use OpenCV functions for rotation and scaling.

In [2]:

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
```

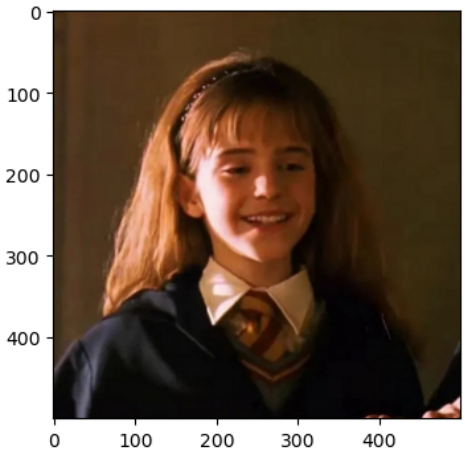
读取图片

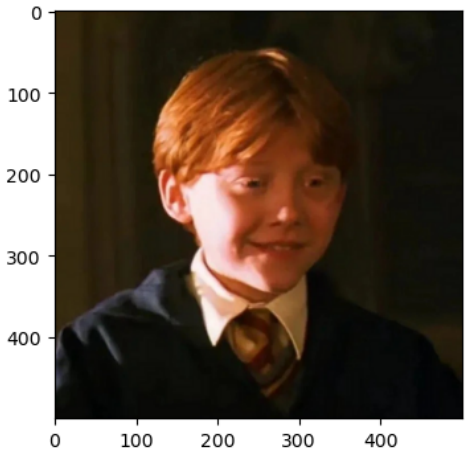
In [3]:

```
image1 = cv2.imread('f.jpg')[:, :, ::-1]
image2 = cv2.imread('m.jpg')[:, :, ::-1]
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.imshow(image1)
plt.subplot(1, 2, 2)
plt.imshow(image2)
```

Out[3]:

```
<matplotlib.image.AxesImage at 0x2ea692cc130>
```



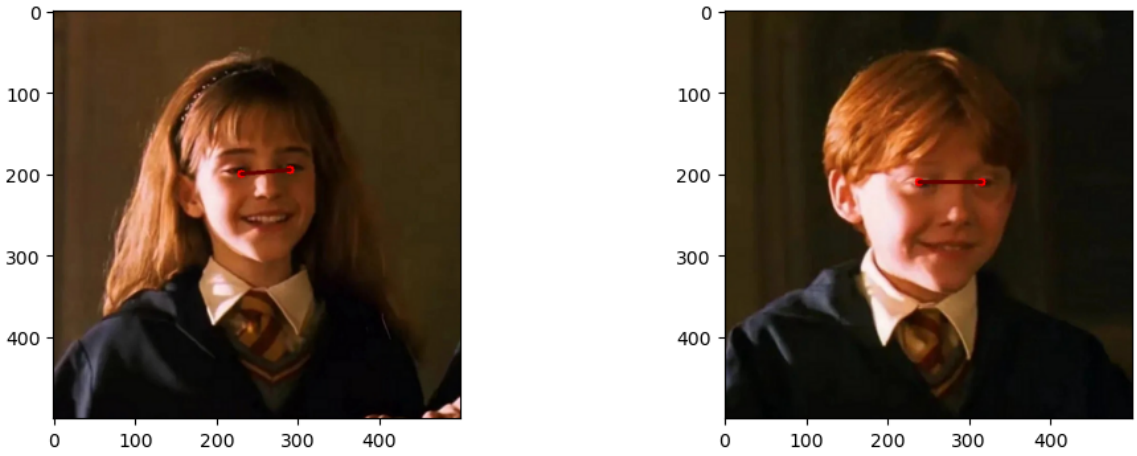


In [4]:

```
# for example, the two points on the first image are:
p11 = [230, 200]
p12 = [290, 195]
# and the corresponding two points on the second image are:
p21 = [238, 210]
p22 = [315, 210]
# TODO: write your own code to calculate the rotation and scaling parameter
image1 = cv2.imread('f.jpg')
image2 = cv2.imread('m.jpg')
cv2.circle(image1, tuple(p11), 5, (0, 0, 255), -1)
cv2.circle(image1, tuple(p12), 5, (0, 0, 255), -1)
cv2.line(image1, tuple(p11), tuple(p12), (0, 0, 100), 4)
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.imshow(image1[:, :, ::-1])

cv2.circle(image2, tuple(p21), 5, (0, 0, 255), -1)
cv2.circle(image2, tuple(p22), 5, (0, 0, 255), -1)
cv2.line(image2, tuple(p21), tuple(p22), (0, 0, 100), 4)
plt.subplot(1, 2, 2)
plt.imshow(image2[:, :, ::-1])
```

Out[4]:

```
<matplotlib.image.AxesImage at 0x2ea6984d640>
```



In [5]:

```
# 计算夹角
# 传入两个点相减以后向量对应列表
def angle_calcu(vec1, vec2):
    v1 = np.array(vec1)
    v2 = np.array(vec2)
    # 计算向量的点积
    dot_product = np.dot(v1, v2)
    # 计算向量的模长
    norm_v1 = np.linalg.norm(v1)
    norm_v2 = np.linalg.norm(v2)
    # 计算夹角（弧度）
    cos_theta = dot_product / (norm_v1 * norm_v2)
    theta = np.arccos(cos_theta)
    # 将弧度转换为角度
    angle = np.degrees(theta)
    print("向量v1和向量v2之间的夹角为：", angle, "度")
    return angle
```

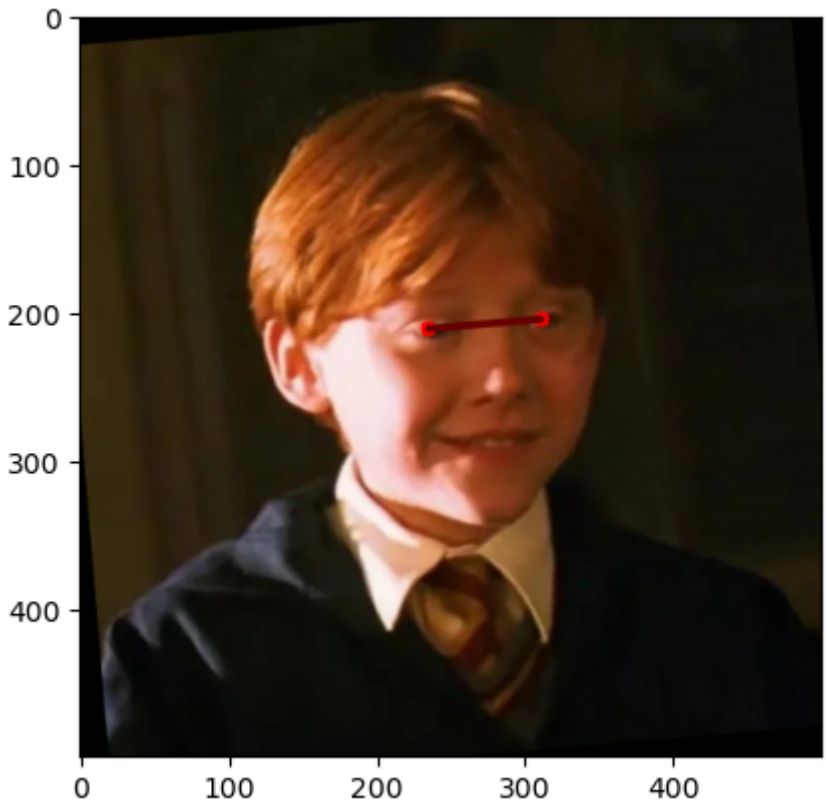
In [6]:

```
# TODO: apply the rotation and scaling on the image
def rotate(img, vec1, vec2):
    # 获取图像尺寸
    height, width = img.shape[:2]
    # 计算旋转矩阵
    M = cv2.getRotationMatrix2D((width / 2, height / 2), angle_calcu(vec1, vec2), 1)
    # 进行旋转变换
    rotated_img = cv2.warpAffine(img, M, (width, height))
    # 显示图像
    return (rotated_img)


# 显示旋转角度以及旋转后的第二张图片
plt.imshow(rotate(image2, [60, -5], [77, 0])[:, :, ::-1])
```

向量v1和向量v2之间的夹角为： 4.76364169072622 度

Out[6]:

<matplotlib.image.AxesImage at 0x2ea6aa32790>



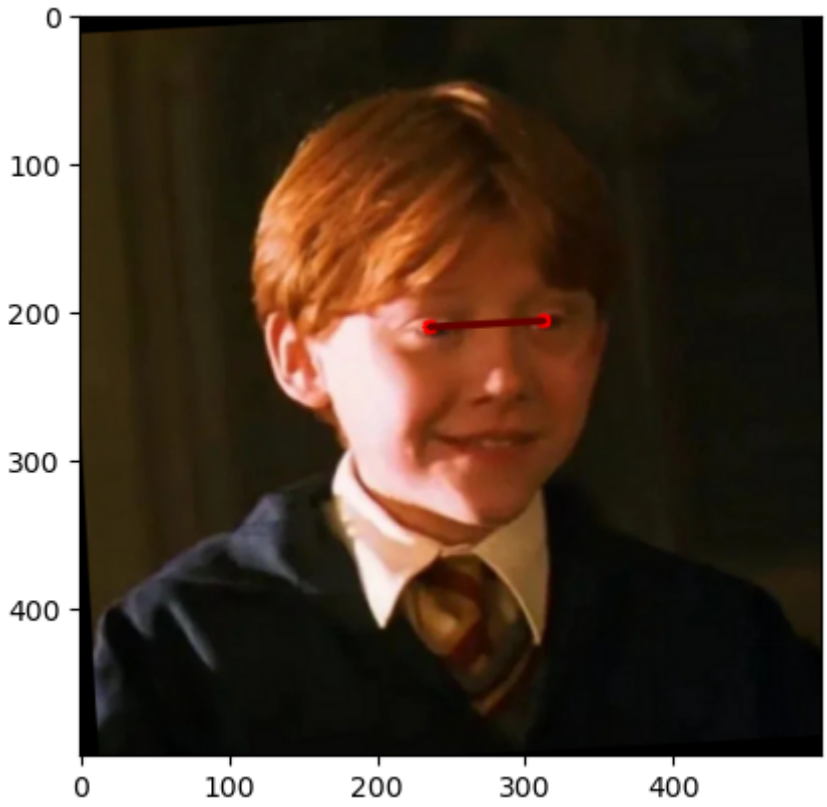
In [7]:

```
# 实现图像缩放，scale_percent为缩放比例
def scaleee(img, scale_percent):
    # 读取图片

    # 获取原始图像的高度和宽度
    height, width = img.shape[:2]

    # 计算缩放后的新高度和宽度
    new_height = int(height * scale_percent / 100)
    new_width = int(width * scale_percent / 100)

    # 缩放图像
    resized_img = cv2.resize(img, (new_width, new_height), interpolation=cv2.INTER_AREA)
    return resized_img


im_test = rotate(image2, [60, -5], [68, -2])[:, :, ::-1]
plt.imshow(im_test)
```

向量v1和向量v2之间的夹角为： 3.078957372829998 度

Out[7]:

<matplotlib.image.AxesImage at 0x2ea6a879160>

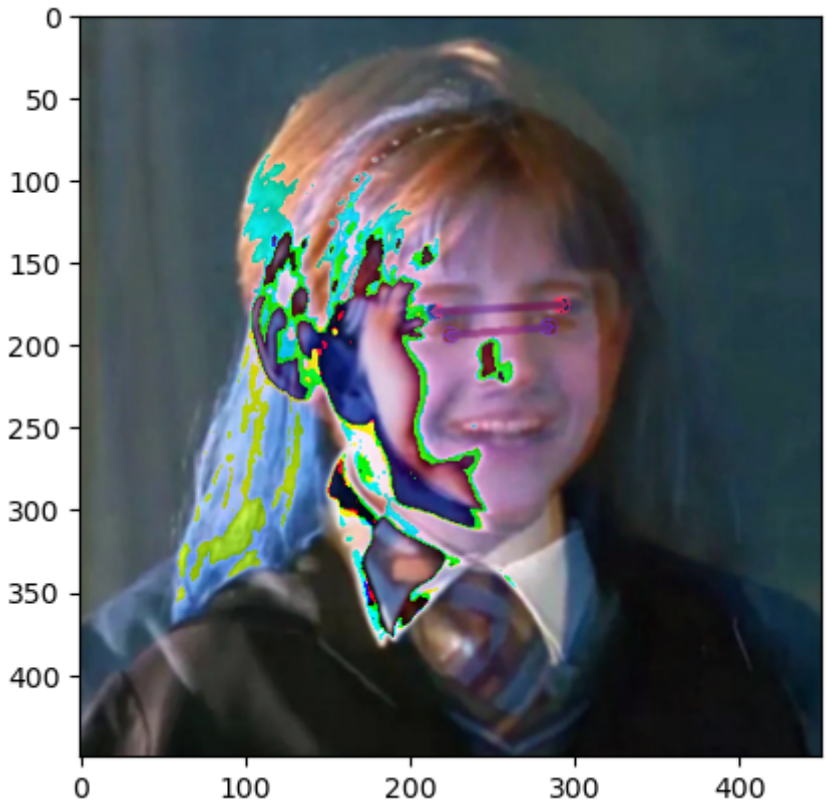


In [8]:

```
# 左上角坐标(x,y)，裁剪区域的宽度和高度(w,h)
def scalee(img, x, y, w, h):
    cropped_img = cv2.getRectSubPix(img, (w, h), (x + w / 2, y + h / 2))
    return cropped_img


imm2 = scalee(image1, 5, 5, 450, 450)
imm1 = scalee(im_test, 20, 30, 450, 450)
plt.imshow(imm2 + imm1)
# 先直接强行堆叠，观看对齐效果
```

Out[8]:

<matplotlib.image.AxesImage at 0x2ea6aa704c0>



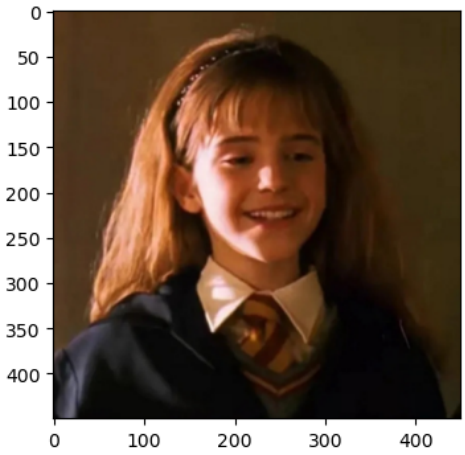
In [84]:

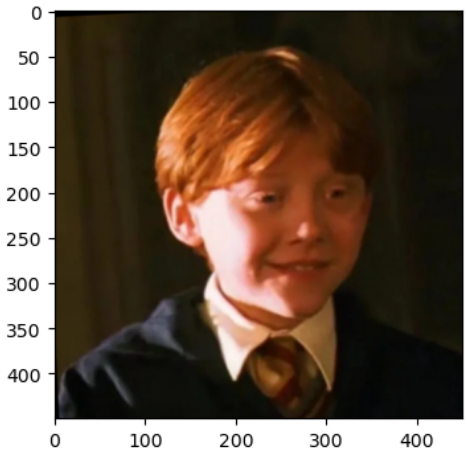
```
# 裁剪
# 由于之前读入的图片被画了线和点，因此重新读入一次
im1 = cv2.imread('f.jpg')[:, :, ::-1]
im2 = cv2.imread('m.jpg')[:, :, ::-1]
im2 = rotate(im2, [60, -5], [68, -2])

# 裁剪为495x495的图片
im2 = scalee(im2, 5, 5, 450, 450)
im1 = scalee(im1, 20, 30, 450, 450)
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.imshow(im1)
plt.subplot(1, 2, 2)
plt.imshow(im2)
```

向量v1和向量v2之间的夹角为： 3.078957372829998 度

Out[84]:

<matplotlib.image.AxesImage at 0x2ea7b7419d0>





2. Make Gaussian filters

You need to make your own Gaussian filters even the OpenCV has provided you with some easy to use functions as follows.

When you make your own Gaussian filter, you have to specify the filter size (usually odd number) and the standard deviation. In OpenCV, if the kernerl size is given but the standard deviation σ is not given, then σ can be determined according to:

$$\sigma = 0.3 * ((ksize - 1) * 0.5 - 1) + 0.8$$

If the σ is given but the kernel size is not given, then the kernel size can be se to 6σ or 8σ (usually an odd number), as we have discussed in the lecture that $[-3\sigma, 3\sigma]$ of a Gaussian function captures 99.7% of the energy.

This blog (https://www.cnblogs.com/shine-lee/p/9671253.html) gives a detailed discussion on the selection of kernel size and σ.

Please finish the following function to generate your Gaussian filter:

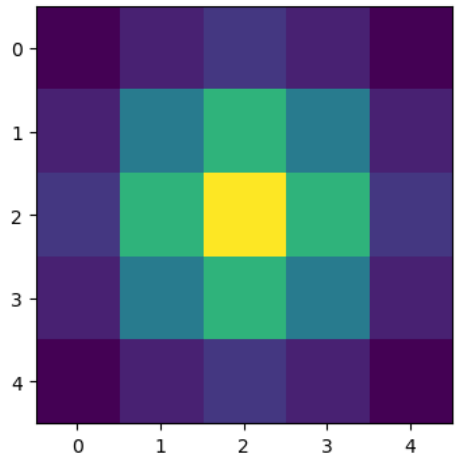
NOTE!!!! The sum of the kernel weights should be 1.

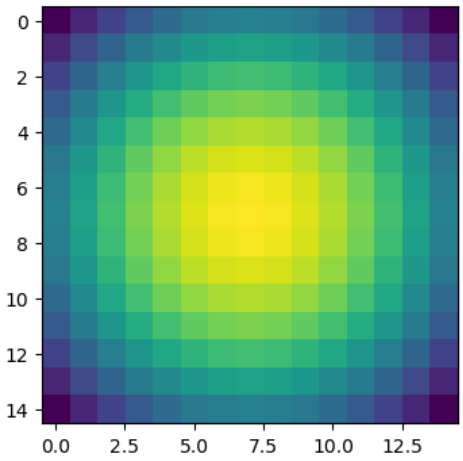
In [85]:

```
# ksize为卷积核大小，sigma为标准差
def MakeGaussianFilter(ksize=None, sigma=None):
    if ksize is None:
        ksize = (int(6 * sigma), int(6 * sigma))
    if sigma is None:
        sigma = 0.3 * ((ksize[0] - 1) * 0.5 - 1) + 0.8
    # 创建全零矩阵表示初始卷积核，大小为ksize
    kernel = np.zeros(ksize, dtype=np.float32)
    # 遍历该卷积核，给每个元素赋值，值服从高斯分布
    for i in range(ksize[0]):
        for j in range(ksize[1]):
            kernel[i, j] = np.exp(
                -(i - ksize[0] // 2) ** 2 / (2 * sigma ** 2) - (j - ksize[1] // 2) ** 2 / (2 *
    kernel /= kernel.sum()
    return kernel


# 显示用到的25x25，标准差为10的高斯滤波器
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.imshow(MakeGaussianFilter((5, 5)))
plt.subplot(1, 2, 2)
plt.imshow(MakeGaussianFilter((15, 15), 10.0))
```

Out[85]:

```
<matplotlib.image.AxesImage at 0x2ea7b69e460>
```

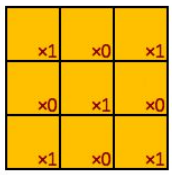




3. Padding your images

We will use zero-padding to keep the filtered image having the same size with the input image. The padding should be half size of your filter kernel. For example, the following figure shows a 3x3 filter and a 5x5 input image. With padding on the top, bottom, left and right with 1 row/column zeros (note, the filter size 3 divided by 2, therefore the padding is 1, 3//2=1), the filtered image has the same size of the input (5x5). If your filter size is, for example 21x21, then the padding should be 10 rows/columns of zeros on each of the 4 boarders.

- Convolution with zero-padding of P pixels



0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	1	1	1	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0

2	2	3	1	1
1	4	3	4	1
2	2	4	3	3
1	2	3	4	1
1	2	3	1	1

Output size:$(H-K+1+2P, W-K+1+2P)$

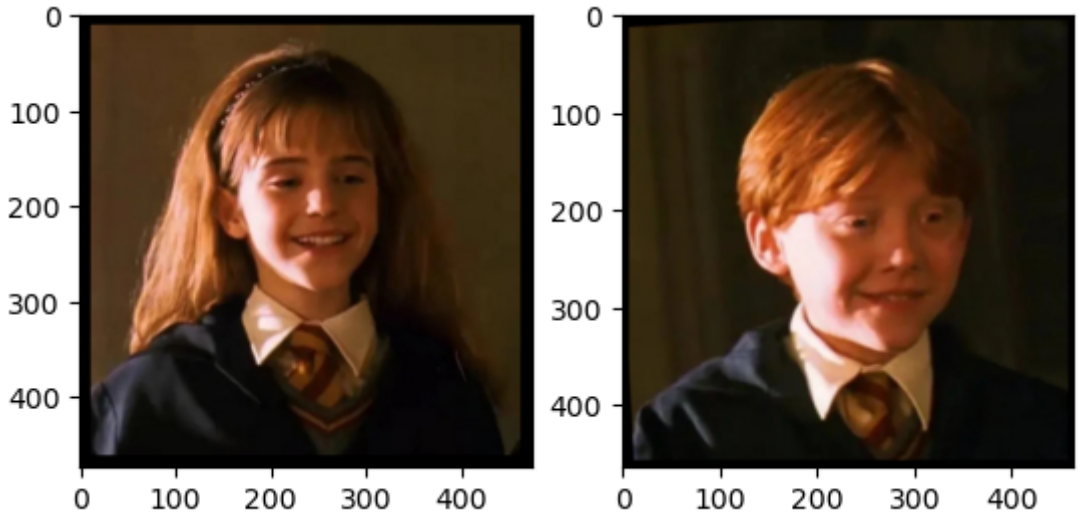
In [86]:

```
print(im1.shape, im2.shape)


def Padding(img, pad):
    # 获取图像尺寸
    h, w = img.shape[:2]
    # 创建一个新的图像，大小为原图像加上padding以后的尺寸
    padding_img = np.pad(img, ((pad, pad), (pad, pad), (0, 0)), mode='constant', constant_values=
    return padding_img


padding_img1 = Padding(im1, 12)
padding_img2 = Padding(im2, 7)
plt.subplot(1, 2, 1)
plt.imshow(padding_img1)
plt.subplot(1, 2, 2)
plt.imshow(padding_img2)
```

```
(450, 450, 3) (450, 450, 3)
```

Out[86]:

```
<matplotlib.image.AxesImage at 0x2ea7d0e4850>
```



In [87]:

```
print(padding_img1.shape, padding_img2.shape)
```

```
(474, 474, 3) (464, 464, 3)
```

4. Convolution

Write your own code for the 2D convolution function. Use a filter to convolve on an image (RGB image).

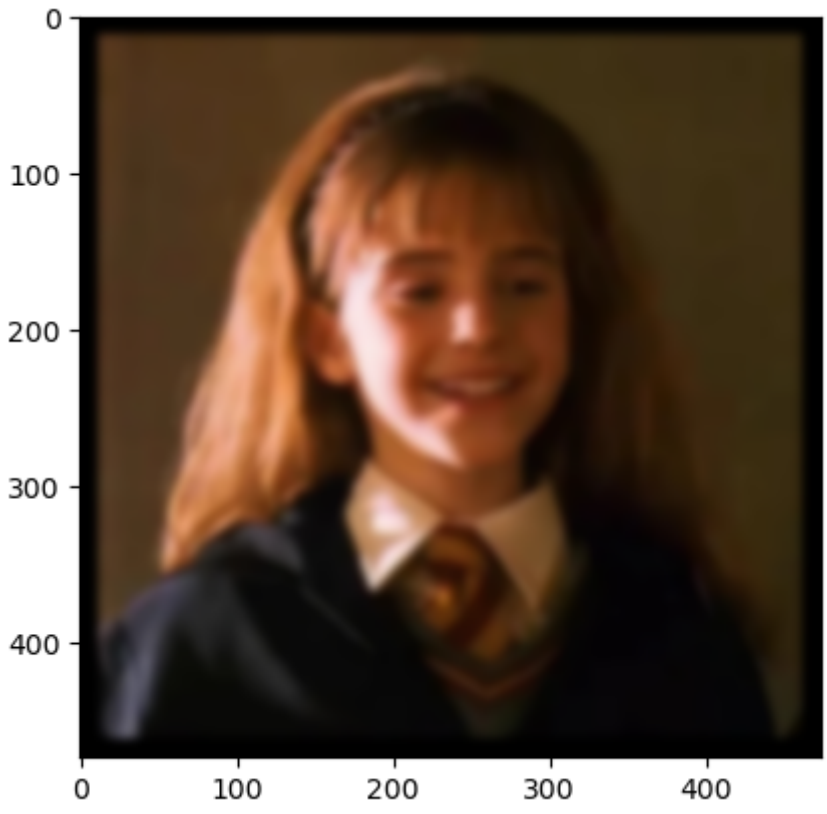
For just this assignment, you are forbidden from using any Numpy, Scipy, OpenCV, or other preimplemented functions for filtering. You are allowed to use basic matrix operations like np.shape, np.zeros, and np.transpose. This limitation will be lifted in future assignments, but for now, you should use for loops or Numpy vectorization to apply a kernel to each pixel in the image.

In [88]:

```
def Convolution2D(img, filt):
    # 获取图像和滤波器的形状
    img_shape = np.array(img.shape)
    filt_shape = np.array(filt.shape)

    # 计算输出图像的形状
    out_shape = img_shape - filt_shape + 1

    # 初始化输出图像
    out = np.zeros(out_shape)

    # 对输入图像进行卷积操作
    for i in range(out_shape[0]):
        for j in range(out_shape[1]):
            # 从输入图像中提取当前窗口
            window = img[i:i + filt_shape[0], j:j + filt_shape[1]]

            # 将窗口和滤波器相乘，然后求和
            out[i, j] = np.sum(window * filt)

    return out


plt.imshow(Convolution2D(padding_img1, MakeGaussianFilter((25, 25), 3.0)))
```

Out[88]:

```
<matplotlib.image.AxesImage at 0x2ea7d5629a0>
```



5. Filter your images and make hybrid image

Make a Gaussian filter to convolve on image A, get the blured image A'. Make another Gaussian filter to convolve on image B, and substract the result from the original B, get the detail image B'. Add A' and B' to get the hybrid image.

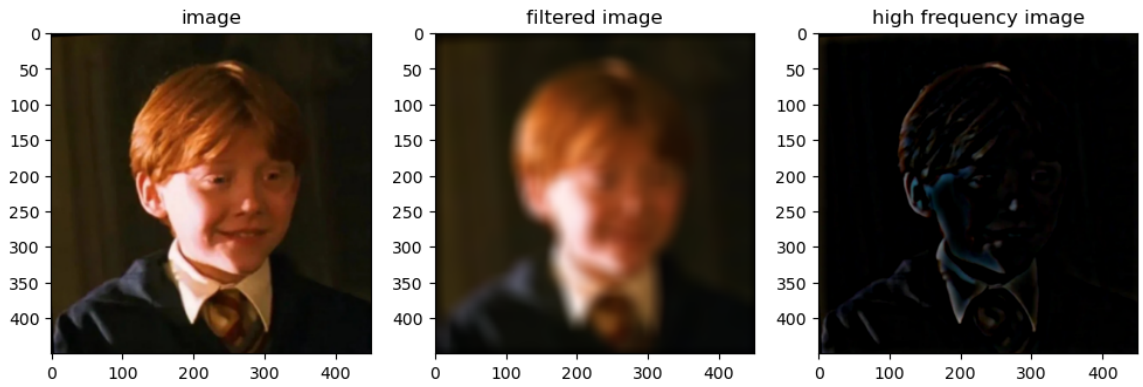
NOTE! You may have to try different values for the standard deviation in Gaussian filters to get satifactory results.

In [89]:

```
# TODO:
img_filtered1 = scalee(Convolution2D(padding_img1, MakeGaussianFilter((25, 25), 15)), 5, 5, 450, 4
img_filtered2 = scalee(Convolution2D(padding_img2, MakeGaussianFilter((25, 25), 25)), 5, 5, 450, 4

scaled_img = (img_filtered2 - np.min(img_filtered2)) / (np.max(img_filtered2) - np.min(img_filter
# 取整数部分
int_img = np.round(scaled_img)
# 转换为uint8类型
uint8_img = int_img.astype(np.uint8)
img_filtered2_subed = cv2.subtract(im2, uint8_img)

plt.figure(figsize=(12, 4))
plt.subplot(1, 3, 1)
plt.title("image")
plt.imshow(im2)
plt.subplot(1, 3, 2)
plt.title("filtered image")
plt.imshow(uint8_img)
plt.subplot(1, 3, 3)
plt.title("high frequency image")
plt.imshow(img_filtered2_subed)
```

Out[89]:

<matplotlib.image.AxesImage at 0x2ea7c7e3b20>



6. Make Gaussian pyramid to show your hybrid image

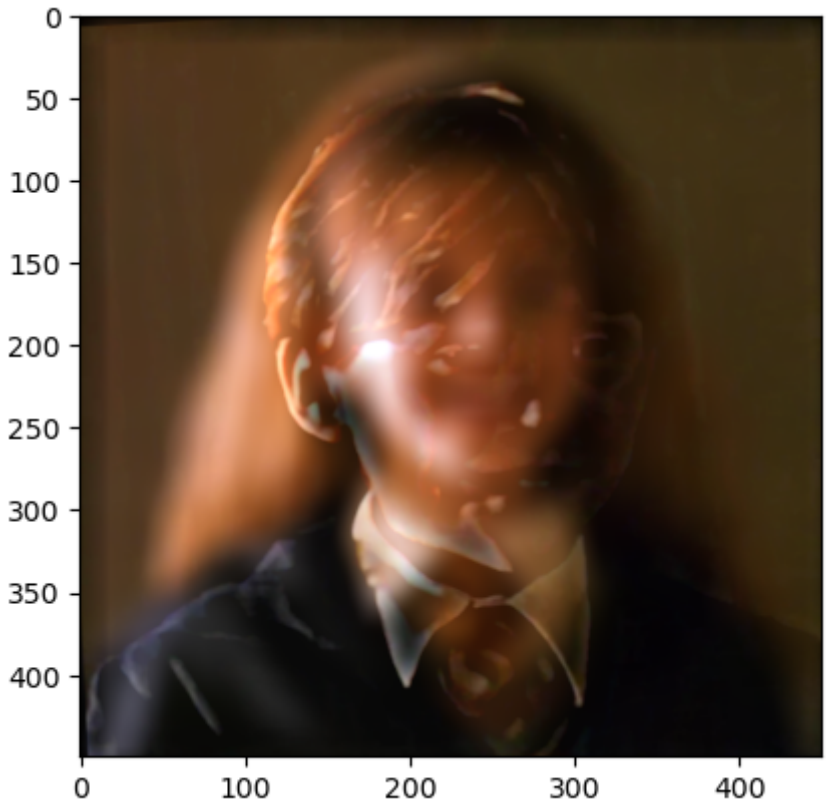
Use Gaussian filter to build up a Gaussian pyramid. The lower level image in the pyramid should exhibit high frequency information (details); while the higher level image in the pyramid should exhibit low frequency information.

In [90]:

```
# TODO:
hybrid_img = cv2.add(img_filtered1, img_filtered2_subed)
plt.imshow(hybrid_img)
```

Out[90]:

```
<matplotlib.image.AxesImage at 0x2ea7ca97880>
```



In [91]:

```
def plott(img, step):
    rows, cols, _ = img.shape
    Rows, Cols = (rows - 1 + step) // step, (cols - 1 + step) // step
    result = np.zeros((Rows, Cols, _), dtype=np.uint8)
    for k in range(_):
        for i in range(Rows):
            for j in range(Cols):
                result[i, j, k] = img[i * step - step + 1, j * step - step + 1, k]
    return result


def plottt(img, n=4, sep=5):
    img_lis = [img]
    rows, cols, _ = img.shape
    for i in range(n):
        img = plott(img, 2)
        img_lis.append(img)
        cols += sep + img.shape[1]
        canvas = np.ones((rows, cols, _), dtype=np.uint8) * 255
        left = 0
    for im in img_lis:
        row, col = im.shape[0], im.shape[1]
        canvas[rows - row:rows, left:left + col, :] = im
        left += sep + col
    return canvas


plt.figure(figsize=(10, 4))
plt.xticks([])
plt.yticks([])
plt.axis('off')
canvas = plottt(hybrid_img, 4, 5)
plt.imshow(canvas)
```

Out[91]:

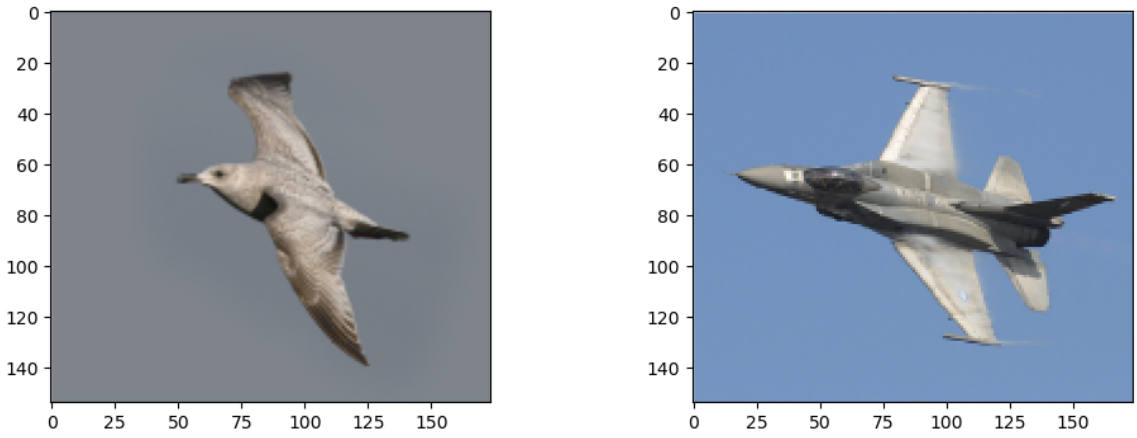
<matplotlib.image.AxesImage at 0x2ea7c9f4c10>

In [92]:

```
# 裁剪
# 由于之前读入的图片被画了线和点，因此重新读入一次
im1 = cv2.imread('bird.png')[:, :, ::-1]
im2 = cv2.imread('plane.png')[:, :, ::-1]

# 裁剪为495x495的图片
#im2 = scalee(im2, 5, 5, 450, 450)
#im1 = scalee(im1, 20, 30, 450, 450)
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.imshow(im1)
plt.subplot(1, 2, 2)
plt.imshow(im2)
```

Out[92]:

```
<matplotlib.image.AxesImage at 0x2ea7c8b5700>
```



In [93]:

```
print(im1.shape, im2.shape)
```

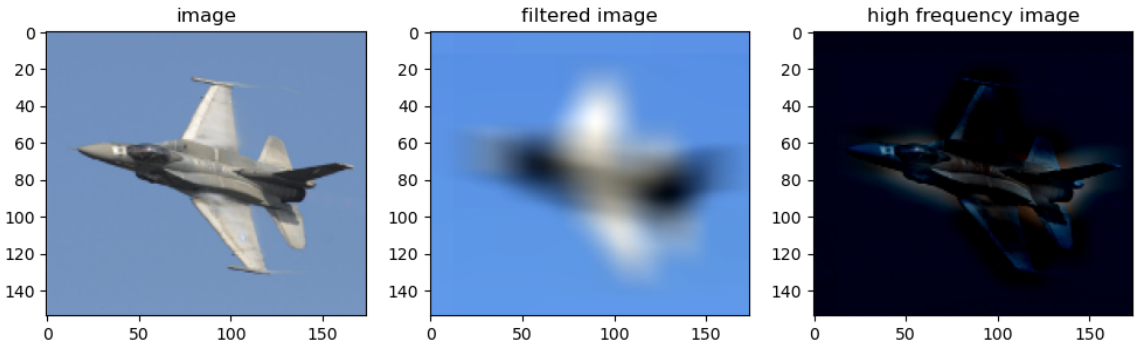
```
(154, 174, 3) (154, 174, 3)
```

In [94]:

```
img_filtered1 = Convolution2D(im1, MakeGaussianFilter((25, 25), 15.0))
img_filtered2 = Convolution2D(im2, MakeGaussianFilter((25, 25), 25.0))
scaled_img = (img_filtered2 - np.min(img_filtered2)) / (np.max(img_filtered2) - np.min(img_filter
# 取整数部分
int_img = np.round(scaled_img)
# 转换为uint8类型
uint8_img = int_img.astype(np.uint8)
img_filtered2_subed = cv2.subtract(im2, uint8_img)

plt.figure(figsize=(12, 4))
plt.subplot(1, 3, 1)
plt.title("image")
plt.imshow(im2)
plt.subplot(1, 3, 2)
plt.title("filtered image")
plt.imshow(uint8_img)
plt.subplot(1, 3, 3)
plt.title("high frequency image")
plt.imshow(img_filtered2_subed[:, :, ::-1])
```

Out[94]:

<matplotlib.image.AxesImage at 0x2ea7cc0db50>

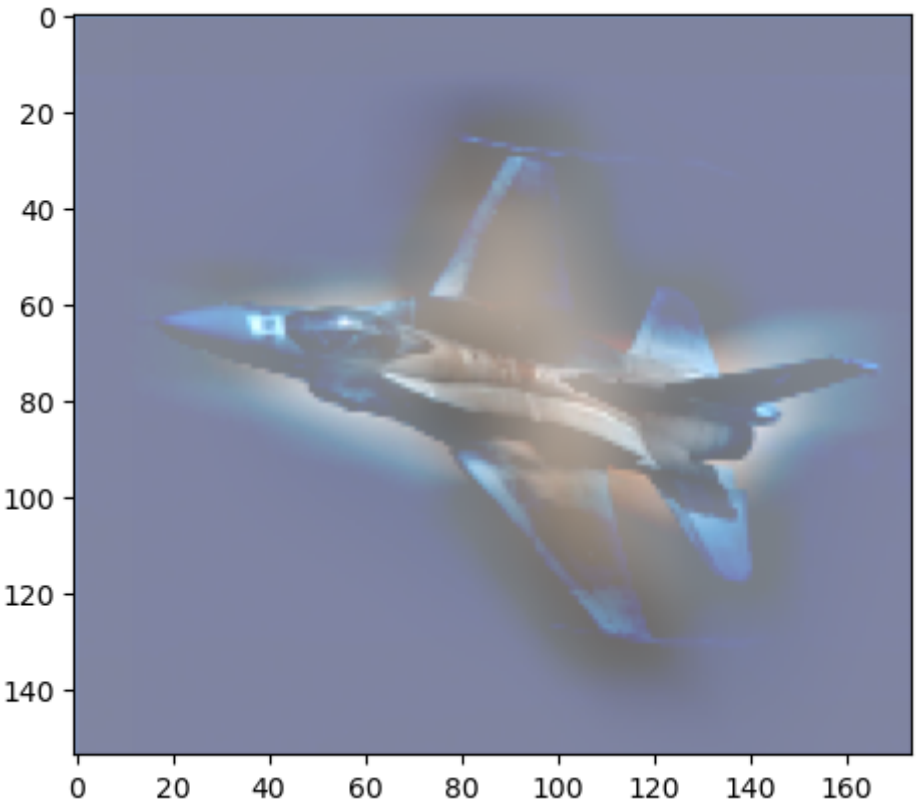


In [95]:

```
hybrid_img = cv2.add(img_filtered1, img_filtered2_subed[:, :, ::-1])
plt.imshow(hybrid_img)
```

Out[95]:

<matplotlib.image.AxesImage at 0x2ea7cc7df40>

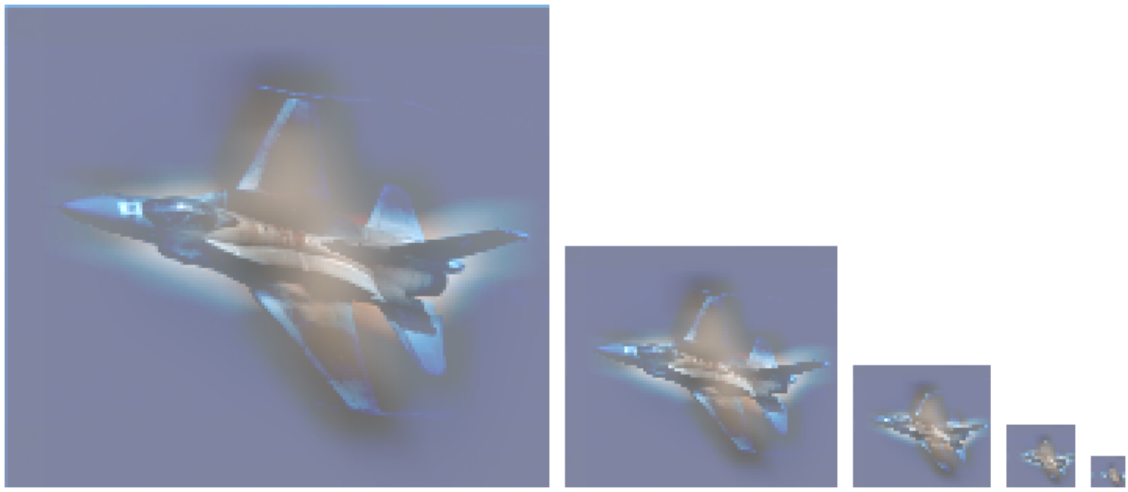


In [96]:

```
plt.figure(figsize=(10, 4))
plt.xticks([])
plt.yticks([])
plt.axis('off')
canvas = plottt(hybrid_img, 4, 5)
plt.imshow(canvas)
```

Out[96]:

```
<matplotlib.image.AxesImage at 0x2ea7ce193d0>
```



References

0. The origianl authors' project page. http://olivalab.mit.edu/hybridimage.htm (http://olivalab.mit.edu/hybridimage.htm)
1. A Tutorial. https://jeremykun.com/2014/09/29/hybrid-images/ (https://jeremykun.com/2014/09/29/hybrid-images/)
2. Chinese blog， matlab code, with alignment。 https://blog.csdn.net/breeze_blows/article/details/102962559 (https://blog.csdn.net/breeze_blows/article/details/102962559)
3. Matlab code. https://github.com/coldmanck/Image-Filtering-and-Hybrid-Images (https://github.com/coldmanck/Image-Filtering-and-Hybrid-Images)
4. Soton University. http://comp3204.ecs.soton.ac.uk/cw/coursework2.html (http://comp3204.ecs.soton.ac.uk/cw/coursework2.html)
5. Brown University. http://cs.brown.edu/courses/cs143/2011/ (http://cs.brown.edu/courses/cs143/2011/) and the project http://cs.brown.edu/courses/cs143/2011/proj1/ (http://cs.brown.edu/courses/cs143/2011/proj1/)
6. Python implementation with GUI. https://github.com/ReynoldZhao/Hybrid_Images (https://github.com/ReynoldZhao/Hybrid_Images)
7. George Washington University, student's homework. https://blog.csdn.net/Sengo_GWU/article/details/79336511 (https://blog.csdn.net/Sengo_GWU/article/details/79336511)
8. Georgia Institute of Technology, student's homework. https://github.com/all4win/Computer_Vision_Proj1_Image_Filtering_and_Hybrid_Images (https://github.com/all4win/Computer_Vision_Proj1_Image_Filtering_and_Hybrid_Images) and

https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj1/html/jwei74/index.html (https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj1/html/jwei74/index.html)
9. 电子科大的学生作业. https://blog.csdn.net/sinat_41942180/article/details/107972994 (https://blog.csdn.net/sinat_41942180/article/details/107972994)
10. 比较详细的中文博客文章，Matlab代码. https://blog.csdn.net/weixin_45901986/article/details/104823057 (https://blog.csdn.net/weixin_45901986/article/details/104823057)
11. 英国南安普顿大学的博士研究生的知乎文章. https://zhuanlan.zhihu.com/p/106619097 (https://zhuanlan.zhihu.com/p/106619097)
12. Washington University in St. Louis. Student's project. https://sites.google.com/site/cse559acomputervisionprojects/home/project-1-hybrid-images (https://sites.google.com/site/cse559acomputervisionprojects/home/project-1-hybrid-images)