In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.ndimage import convolve
from scipy import signal

# create a 2D Gaussian kernl
def create_2d_gaussian(size=9, std=1.5):
    gaussian_1d = signal.gaussian(size,std=std)
    gaussian_2d = np.outer(gaussian_1d, gaussian_1d)
    gaussian_2d = gaussian_2d/(gaussian_2d.sum())
    return gaussian_2d

# normalize image between 0 and 1
def normalize_img(img):
    normalized = (img - img.min())/(img.max() - img.min())
    return normalized

# stack visualization
def visualize_stack(in_stack, levels, title):
    fig, ax = plt.subplots(nrows=2, ncols=3, figsize=(20,10))
    ax = ax.flatten()
    for i in range(levels):
        ax[i].imshow(in_stack[i], cmap='gray')
        ax[i].axis('off')
    plt.suptitle(title)
    plt.show()

# build Gaussian and Laplacian stack of height levels (img is single channel)
def gaussian_and_laplacian_stack(img, levels):
    gaussian = create_2d_gaussian(size=17, std=3)
    gaussian_stack = []
    img_gaussian = img.copy()
    for i in range(levels):
        if i == 0:
            gaussian_stack = [img_gaussian]
        else:
            gaussian_stack.append(convolve(gaussian_stack[-1], gaussian, mode='r

    laplacian_stack = []
    for i in range(len(gaussian_stack)): # building the laplacian stack
        if i == len(gaussian_stack)-1:
            laplacian_stack.append(gaussian_stack[i]) # appending the last gauss
        else:
            laplacian_stack.append(gaussian_stack[i]-gaussian_stack[i+1]) # taki
    return (gaussian_stack, laplacian_stack)

# collapse Laplacian stack into single image
def collapse_laplacian_stack(laplacian_stack):
    for i in range(len(laplacian_stack)): # element-wise addition over the stack
        if i==0:
            img_laplacian = laplacian_stack[0]
        else:
            img_laplacian = np.add(img_laplacian,laplacian_stack[i])
    return img_laplacian

# create new (blended) stack by combining invidivual stack levels
def create_blended_stack(ls1, ls2, gs):
    blended_stack = []
```

```
    for i in range(len(ls1)):
        blended_stack.append((gs[i]*ls1[i])+(1-gs[i])*(ls2[i])) #(Gk x L1k) + (1
    return blended_stack

# perform image blending
def multires_blending(img1, img2, mask, levels):
    gs_1, ls_1 = gaussian_and_laplacian_stack(img1, levels)
    gs_2, ls_2 = gaussian_and_laplacian_stack(img2, levels)
    gs_m, _    = gaussian_and_laplacian_stack(mask, levels)
    blended_ls = create_blended_stack(ls_1, ls_2, gs_m) # blend
    return collapse_laplacian_stack(blended_ls)

# load images
im1 = plt.imread("a4-hf.png")
im2 = plt.imread("a4-terminator.png")
msk = plt.imread("a4-mask.png")

# create Gaussian and Laplacian stacks
levels     = 6
gs_a, ls_a = gaussian_and_laplacian_stack(im1, levels)
gs_o, ls_o = gaussian_and_laplacian_stack(im2, levels)
gs_m, _    = gaussian_and_laplacian_stack(msk, levels)

# visualize all the stacks
visualize_stack(ls_a, levels, title='laplacian hf')
visualize_stack(ls_o, levels, title='laplacian terminator')

blend = normalize_img(multires_blending(im1, im2, msk, levels))

dpi  = 72.0
height, width = blend.shape
figsize = width/dpi, height/dpi
plt.figure(figsize=figsize)
plt.imshow(blend)
plt.axis("off")
plt.gray()
plt.show()
```

laplacian hf

laplacian terminator