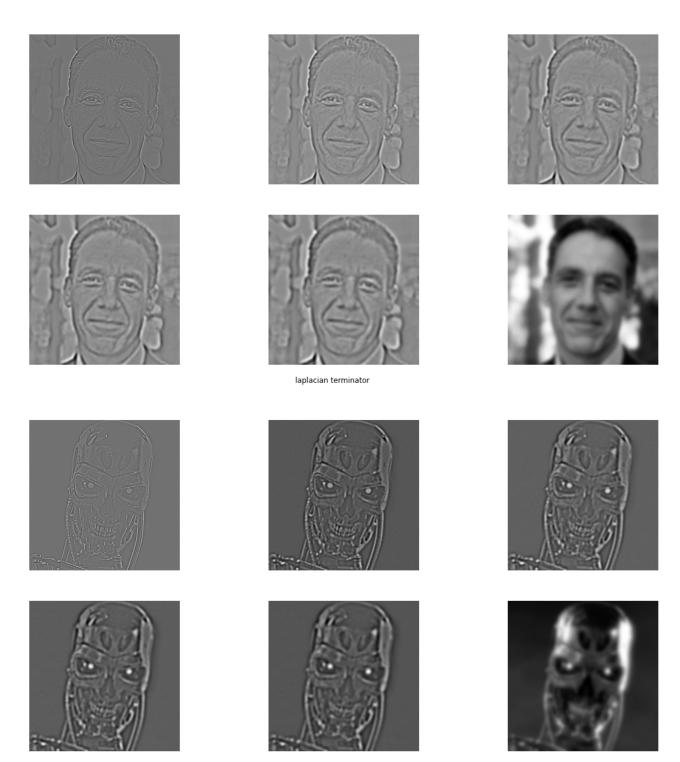
10/2/22, 7:23 PM HW4\_MelissaLicari

```
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]
                     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 = []
```

10/2/22, 7:23 PM HW4\_MelissaLicari

```
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
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()
```



10/2/22, 7:23 PM HW4\_MelissaLicari

