IMAGE PROCESSING
2.ASSIGNMENT REPORT
IMAGE BLENDING

Prepared by: Sertaç Güler
School ID: 21986764

Problem Definition:

Image blending is some of the most useful techniques in Image processing. From the name it mixes 2 or more images seamlessly.However the problem is that when you directly mix images it will preserve edges and the result will not be satisfactory. We can separate edge information from the low information using image pyramids. For this we can do gaussian blurring to remove high frequency components so that the shah function will not overlap and give aliasing effects. We then downsample the image to half resolution. This is done because we want the target image to blend. These pyramids can also be used to detect different scale objects in an image.We use Gaussian pyramids to create Laplacian pyramids which subtracts from the low level pyramid the high level pyramid and stores the edge information this way. Finally we collapse 2 images' laplacian pyramids using the mask image provided by the user.

1) Taking Mask Region From the User:

I read 2 images img1 will be the main image and img2 will be the extra part I will blend. In case they are not the same size I resize them. I use the ROI (Region of Interest) function of the CV2 library to let users choose the mask region. Mask will have 255(white) for the region mask and 0(black) for the not chosen region.

```
# Load images
img1 = cv2.imread('madrid1.jpg')
img2 = cv2.imread('barca1.jpg')
print("stars shape:", img1.shape)
print("planets shape:", img2.shape)
img2 = cv2.resize(img2, (img1.shape[1], img1.shape[0]))
#Select ROI
roi = cv2.selectROI(img2)
cv2.destroyAllWindows()

#Extract selected region
x, y, w, h = roi
mask = np.zeros(img2.shape[:2], dtype=np.uint8)
mask[y:y+h, x:x+w] = 255

#Display mask
cv2.imshow("Mask", mask)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2) Building Gaussian Pyramids:

I use 2 functions. Downsample function will blur the image to remove high frequency components using Gaussian blur. Then using resize I will obtain a half resolution image.

```
def downsample(img):
    blurred = cv2.GaussianBlur(img, (5, 5), 0)
    return cv2.resize(blurred, (blurred.shape[1] // 2, blurred.shape[0] // 2))
```

BuildGaussianPyramids function will take the original image and use the downsample function to create gaussian pyramids with give levels value which specifies the layer number of the pyramid.

```
def buildGaussianPyramid(img, levels):
    pyramid = [img]
    for _ in range(levels - 1):
        img = downsample(img)
        pyramid.append(img)
    return pyramid
```

3) Building Laplacian Pyramids:

I use the results from gaussian pyramids of the images and simply subtract the upper level (lower resolution) images from the one level down image by upsampling the downsampled image. This way I will obtain edge information which will be useful for collapsing and getting the final blended image.

```
def buildLaplacianPyramid(gaussPyramid):
    lapPyramid = []
    for i in range(len(gaussPyramid) - 1):
        size = (gaussPyramid[i].shape[1], gaussPyramid[i].shape[0])
        upsampled = upsample(gaussPyramid[i + 1], size)
        laplacian = cv2.subtract(gaussPyramid[i], upsampled)
        lapPyramid.append(laplacian)
    lapPyramid.append(gaussPyramid[-1])
    return lapPyramid
```

```
def upsample(img, size):
    return cv2.resize(img, size)
```

4) Blending Images:

I use the masking gaussian pyramid with same level to create a blendedLayer which will provide the blended pyramid image for that level. For each level I apply the same blending function.

```
blendedPyramid = []
for i in range(levels):
    L1 = lapPyramid1[i]
    L2 = lapPyramid2[i]
    maskLayer = maskGaussPyramid[i]

    #Normalize mask for blending and expand dimensions to match color image
    maskNorm = maskLayer.astype(np.float32) / 255.0
    maskNorm = np.expand_dims(maskNorm, axis=2)  #Expanding to add a third dimension
    maskNorm = np.repeat(maskNorm, 3, axis=2)   #Repeating the mask for 3 channels

    #Blend
    blendedLayer = L1 * maskNorm + L2 * (1 - maskNorm)
    blendedPyramid.append(blendedLayer)
```

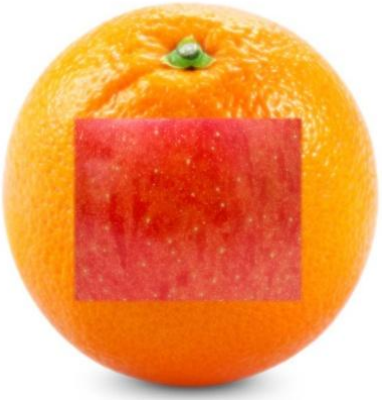
5) Collapsing Images:

To collapse the images in the pyramid levels I start from the top of the pyramid by upsampling the downsampled images and adding the blended layer.

```
def collapsePyramid(pyramid):
    img = pyramid[-1]
    for i in range(len(pyramid) - 2, -1, -1):
        img = upsample(img, (pyramid[i].shape[1], pyramid[i].shape[0]))
        img = cv2.add(pyramid[i], img)
    return img
```

RESULTS:

I tested my program using 5 pairs of different images. I continued to increase the level of downsampling until it can not be downsampled again. For some images the maximum level was 8 and for some it was 11. Because I had different resolution images. I will examine some characteristics of my results using specific image to image comparison.



level = 2

level = 5

level = 7

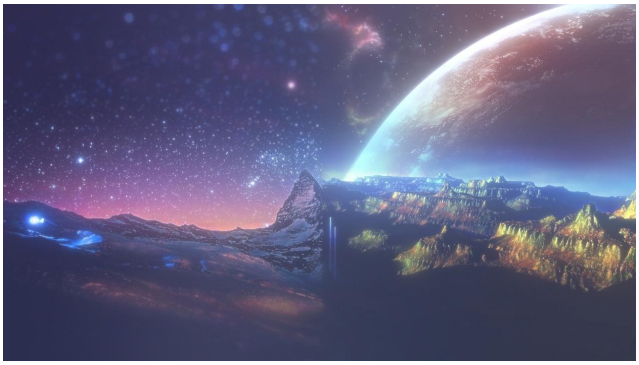
level = 9

For this experiment I tried to blend an apple into and orange image. For the level=2 case I got very discretized image which can be understood simply. I think best cases were when level = 5 and level = 7. When I increased the level to 9 it turned into a whole different fruit mostly resembling to an orange but some wrinkles of an apple.It is also washed out because of mixing the background white into whole image.We can conclude that when we increase the pyramid level it will have some kind of

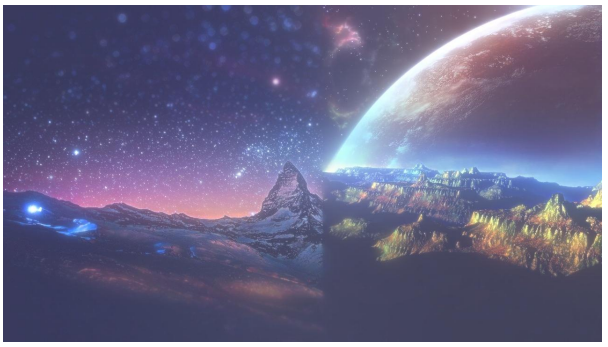
overall mixture expanded through the image. Like in the level=9 we see some background colors like a mix of colors of apple and orange. Level=7 is the most seamless that can be accepted. However for level = 5 we also got a satisfactory result with less mixed values.

level = 5

level = 7



level = 9



level = 10

For this experiment I tried to blend one side of one image to other side of the other image. For level= 5 we have a distinctive edge between. As level increases the edge starts to vanish slowly. However the blurriness increases as well because of high mixture.

Note: I have uploaded the other results with blended images names and levels used.