BBM 415 - AIN 432: Image Processing Lab. Assignment 2 Report

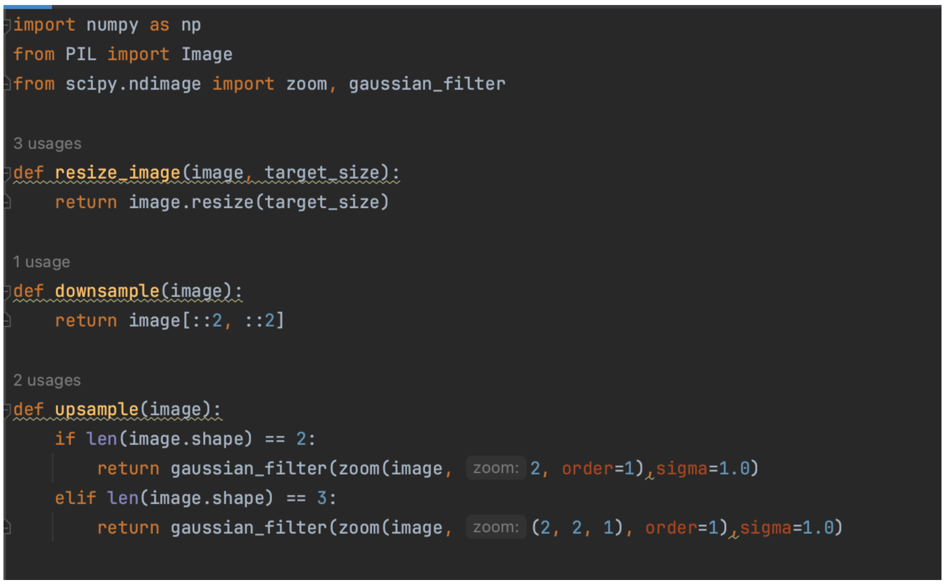
Fall 2023   Instructor: Aydın Kaya   TA: Burçak Asal

Yağız Buğra Boz - 12 Aralık 2023

Brief Overview of The Problem

The goal of this assignment is to obtain a blended image from two different image. The most important thing for image blending methods to blend images seamlessly. In other words for an successful image blending method, seams where images or image regions are stitched must be invisible.

Details of The Approach - Algorithm Explanation



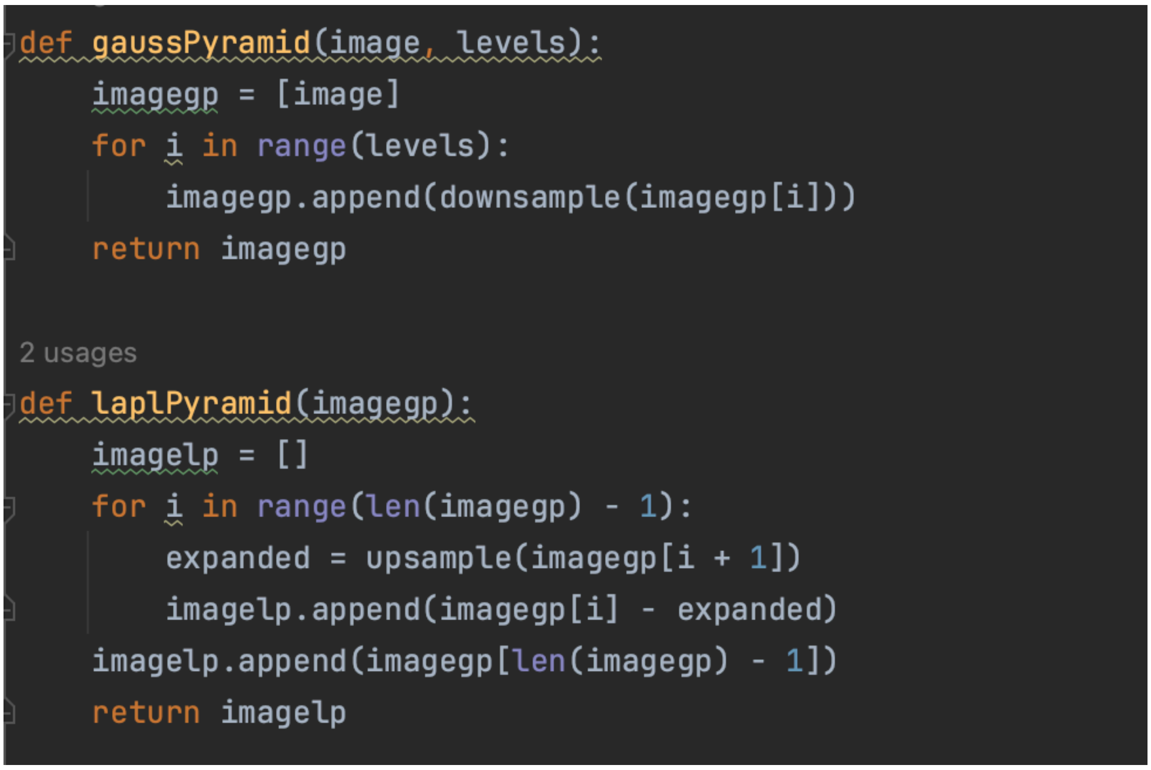
```
import numpy as np
from PIL import Image
from scipy.ndimage import zoom, gaussian_filter

3 usages
def resize_image(image, target_size):
    return image.resize(target_size)

1 usage
def downsample(image):
    return image[::2, ::2]

2 usages
def upsample(image):
    if len(image.shape) == 2:
        return gaussian_filter(zoom(image,  zoom: 2, order=1),sigma=1.0)
    elif len(image.shape) == 3:
        return gaussian_filter(zoom(image,  zoom: (2, 2, 1), order=1),sigma=1.0)
```

1.**Part:** The necessary libraries was imported.

2.Part (resize_image): A function is defined that takes an image and a target size as input and returns the resized image. This method is used to make images the same size.

Part 3 (downsample): A function is defined that takes an image as input and performs downsampling by taking one of every two pixel along both dimensions.

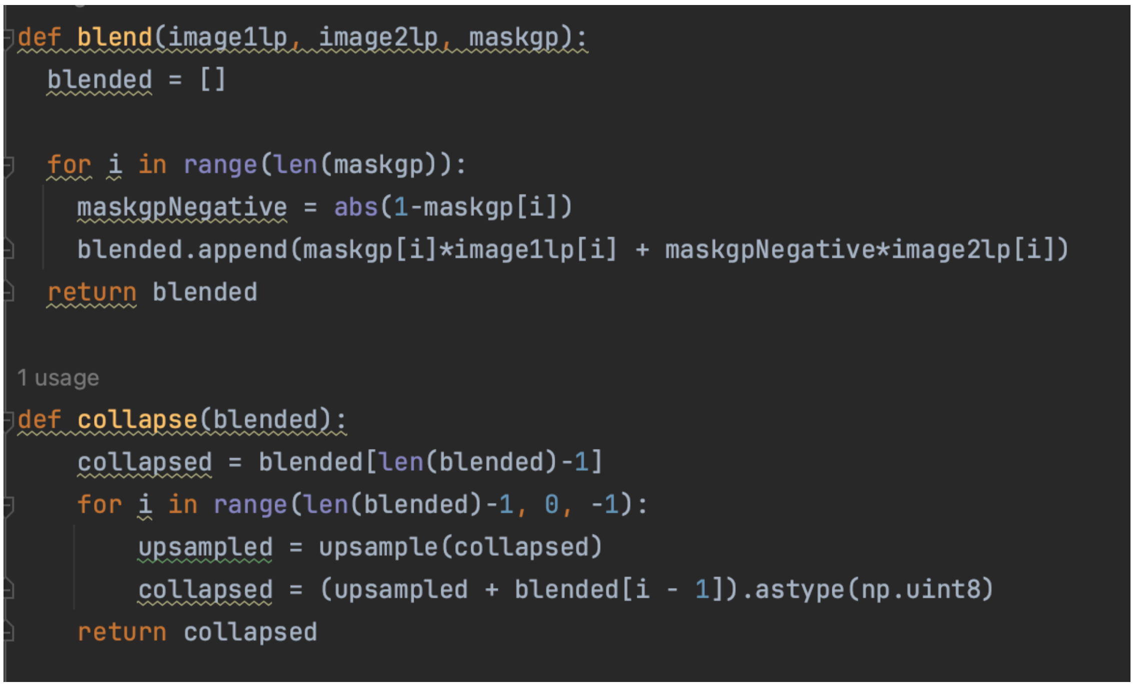
Part 4 (upsample): A function is defined that takes an image as input, checks whether the image is 2D or 3D (to prevent errors about dimensions) then zoom and apply Gaussian filtering with specified sigma.



```
def gaussPyramid(image, levels):
    imagegp = [image]
    for i in range(levels):
        imagegp.append(downsample(imagegp[i]))
    return imagegp

2 usages
def laplPyramid(imagegp):
    imagelp = []
    for i in range(len(imagegp) - 1):
        expanded = upsample(imagegp[i + 1])
        imagelp.append(imagegp[i] - expanded)
    imagelp.append(imagegp[len(imagegp) - 1])
    return imagelp
```

Part 1 (gaussPyramid): "imagegp = [image]": Initializes a list called "imagegp"with the original image as its first element. "for i in range(levels)": This is a loop that iterates levels times. In each iteration, it appends a downsampled version of the image at the current level "i" to the "imagegp" with "imagegp.append(downsample(imagegp[i]))".

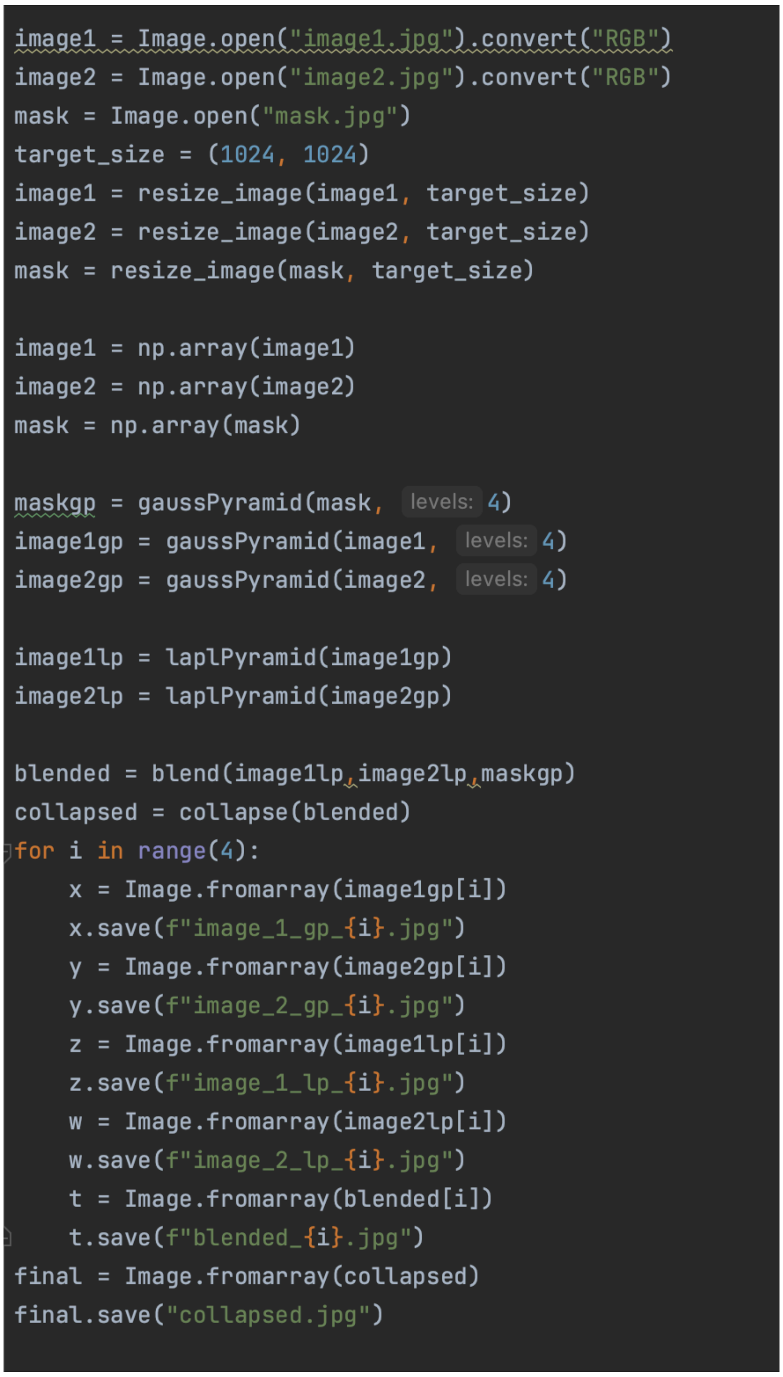
Part 2 (laplPyramid): "for i in range(len(imagegp) - 1):"is a loop that iterates over the levels of the Gaussian pyramid. Then "expanded = upsample(imagegp[i + 1])" computes the upsampled version of the next level of the Gaussian pyramid. Then calculates the difference between the current level and the upsampled version and appends it to "imagelp". Then appends the last level of the Gaussian pyramid to "imagelp".



```
def blend(image1lp, image2lp, maskgp):
  blended = []

  for i in range(len(maskgp)):
    maskgpNegative = abs(1-maskgp[i])
    blended.append(maskgp[i]*image1lp[i] + maskgpNegative*image2lp[i])
  return blended

1 usage
def collapse(blended):
    collapsed = blended[len(blended)-1]
    for i in range(len(blended)-1, 0, -1):
        upsampled = upsample(collapsed)
        collapsed = (upsampled + blended[i - 1]).astype(np.uint8)
    return collapsed
```

Part 1 (blend): Define a function called blend that takes two Laplacian pyramids "image1lp and image2lp" and a mask Gaussian pyramid "maskgp". Initialize an empty list called blended to store the blended Laplacian pyramid. Iterate over the levels of the mask Gaussian pyramid using a for loop. Calculate the negative of the mask value by subtracting each value from 1. Compute the given formula of the Laplacian pyramids using the mask values. It combines the details of both images based on the mask. Return the list containing the blended Laplacian pyramid.

Part 2 (collapse): Define a function called collapse that takes a blended Laplacian pyramid (blended) as an input parameter. Initialize the collapsed image with the last level of the blended Laplacian pyramid. Iterate over the levels of the blended Laplacian pyramid in reverse order. Compute the upsampled version of the collapsed image. Add the upsampled image to the previous level of the blended Laplacian pyramid. Return the collapsed image.



```
image1 = Image.open("image1.jpg").convert("RGB")
image2 = Image.open("image2.jpg").convert("RGB")
mask = Image.open("mask.jpg")
target_size = (1024, 1024)
image1 = resize_image(image1, target_size)
image2 = resize_image(image2, target_size)
mask = resize_image(mask, target_size)

image1 = np.array(image1)
image2 = np.array(image2)
mask = np.array(mask)

maskgp = gaussPyramid(mask,  levels: 4)
image1gp = gaussPyramid(image1,  levels: 4)
image2gp = gaussPyramid(image2,  levels: 4)

image1lp = laplPyramid(image1gp)
image2lp = laplPyramid(image2gp)

blended = blend(image1lp,image2lp,maskgp)
collapsed = collapse(blended)
for i in range(4):
    x = Image.fromarray(image1gp[i])
    x.save(f"image_1_gp_{i}.jpg")
    y = Image.fromarray(image2gp[i])
    y.save(f"image_2_gp_{i}.jpg")
    z = Image.fromarray(image1lp[i])
    z.save(f"image_1_lp_{i}.jpg")
    w = Image.fromarray(image2lp[i])
    w.save(f"image_2_lp_{i}.jpg")
    t = Image.fromarray(blended[i])
    t.save(f"blended_{i}.jpg")
final = Image.fromarray(collapsed)
final.save("collapsed.jpg")
```

1- Open and load images and mask.

2- Set the target size for resizing to (x,x).

3- Resize each image to the target size using the "resize_image" function.

4- Convert the images to NumPy arrays for further processing.

5- Create Gaussian pyramids for each of the images (image1, image2, and mask) with four levels using the gaussPyramid function. The resulting pyramids are stored in "maskgp", "image1gp", and "image2gp".

6- Generate Laplacian pyramids for image1 and image2 using the "laplPyramid" function. Store the resulting pyramids in "image1lp" and "image2lp".

7- Blend the Laplacian pyramids of "image1" and "image2" based on the mask using the blend function. The blended Laplacian pyramid is stored in the variable named "blended."

8- Iterate over four levels of the Gaussian and Laplacian pyramids. Save each level of image1gp, image2gp, image1lp, image2lp, and blended as separate JPG files. The file names are formatted with informative prefixes (e.g., "image_1_gp_0.jpg").

9- Reconstruct the final image by collapsing the blended Laplacian pyramid using the collapse function. Convert the resulting collapsed image to a PIL Image. Save the final reconstructed image as "collapsed.jpg."

Experiment 1:

Image 1:

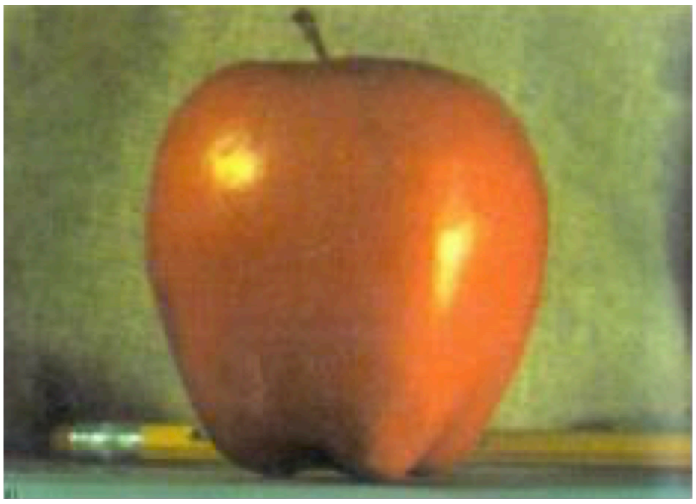
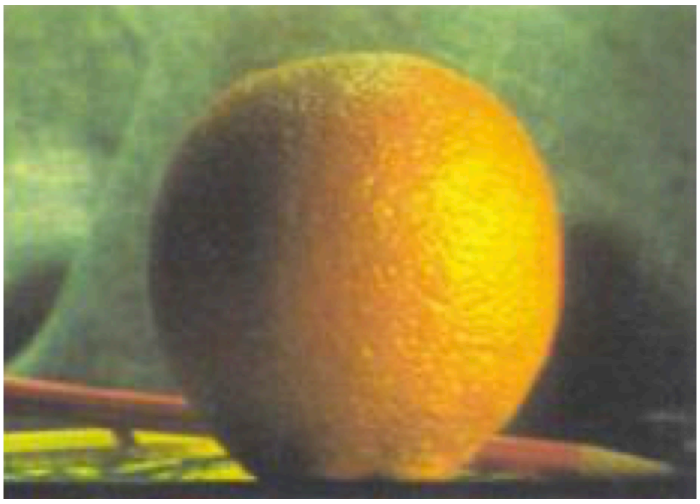
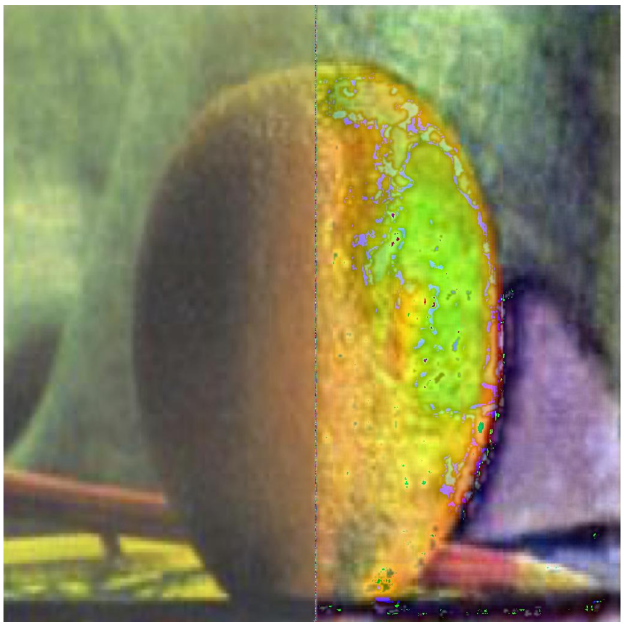


Image2



Final Image:



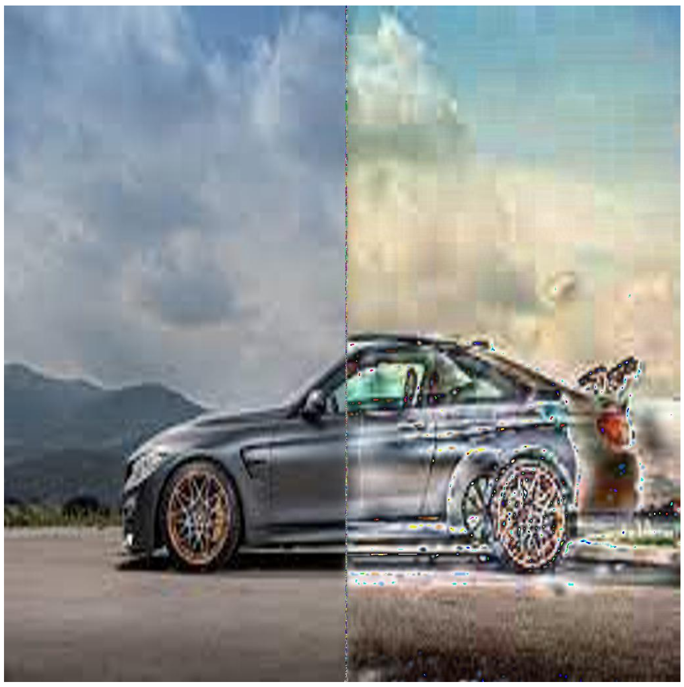
Comment: Although the left side of the image was smooth, there was color distortion on the right side. The cause of the error was not found.

Experiment 2:

Image 1:

Image 2:

Final Image:



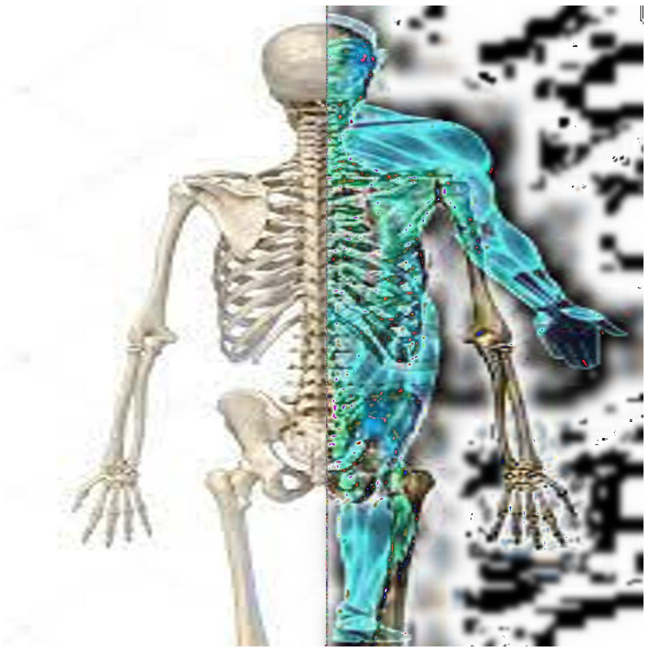
Comment: As in the first experiment, the left side is smooth while the right side is distorted. However, in this experiment, there should be image1 on the right side of the image, but there are both image1 and image2. A situation that can guide you to find this error. but still no solution found.

Experiment 3:

Image 1:

Image 2:

Final Image:



Comment: We again observe the situation experienced in previous experiments. However, in this experiment, we observe that all the red parts in image1 turn blue. There is definitely no smooth transition.

Summary: In this experiments, one side of the image was successful, while the other side suffered color distortion. At the same time, on one side of the final image, there are parts of the image of the other side. It has not been determined whether these problems are related to the color packages used in the code or an error made in a part of the algorithm..