CMPE 362

DIGITAL IMAGE PROCESSING

HOMEWORK 1 REPORT

Instructor: Aslı GENÇTAV
Name - Surname: Neslihan Pelin METİN
Student Number: 71047171244

Introduction

In the first part, we were asked to change the background of an object in an image, and the foreground object should be at the position of the specified top-left corner. To do that, a *combineForegroundBackground* method had to be written with foreground image, foreground mask, background image, and top-left corner parameters.

To implement this method, I took advantage of the two-dimensional property of the images. At first, we removed the mask from the foreground image using the subtract method in the OpenCV library and found the part of the foreground image except for the object. This image is called as *subtracted* in the code. After that, to get the object in the foreground, we removed *subtracted* from the foreground image with the same method and we stored that in a new image that called as *kedi*. Then, we added the image *kedi* to the background image using nested loops where *subtracted* is empty.

```
def combineForegroundBackground(fgImg, fgMask, bgImg, topLeft):
  subtracted = cv2.subtract(fgImg, fgMask)
  subtracted = cv2.cvtColor(subtracted, cv2.COLOR_BGR2RGB)
  kedi = cv2.subtract(fgImg, subtracted)
  kedi = cv2.cvtColor(kedi, cv2.COLOR_BGR2RGB)
  newImg = bgImg
  newImg = cv2.cvtColor(newImg, cv2.COLOR_BGR2RGB)
  x = topLeft[0]
  y = topLeft[1]
  i = 0
  j = 0
  while(x < (len(bgImg)) and i < len(kedi)):
    while(y < (len(bgImg[0])) and j < len(kedi[0])):
      if(any(subtracted[i][j]) == False):
          newImg[x][y] = kedi[i][j]
      y = y + 1
      j = j + 1
    x = x + 1
    i = i + 1
    y = topLeft[1]
    j = 0
  return newImg
```

In the next question, we need to create two new methods whose name are *gaussianSmoothing* and *unsharpMasking*. One should smooth, and the other one should sharpen the image with using Unsharp Masking. In addition, both shall use the Gaussian Smoothing technique. After writing these codes, we need to test these with the background and foreground images, and then combine them with the previous method called *combineForegroundBackground*.

Firstly, to implement *gaussianSmoothing* function, we need to take two parameters which are an image and a sigma value. According to the question, the size of the kernel must be an odd number that is bigger than or equal to 6*sigma. So, in the method we are calculating the kernel size first and later, using *getGaussianKernel* from the OpenCV library with the kernel size we calculated and the given sigma value. By this, our kernel has been created and stored in the object named *filt*. Afterward, we applied this kernel to the given image in both the x and y axis with *sepFilter2D* which is again a method from OpenCV.

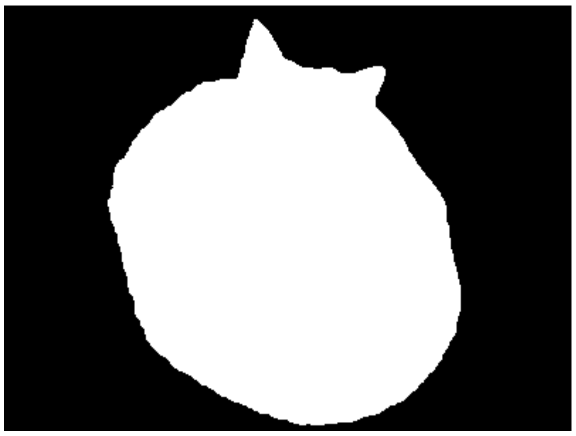
```
def gaussianSmoothing(img, sigma):
  size = 6*sigma
  if(size % 2 == 0):
    size = size + 1
  filt = cv2.getGaussianKernel(ksize = int(size), sigma = sigma)
  img = cv2.sepFilter2D(src=img, ddepth = -1, kernelX =  filt, kernelY = filt)
  return img
```

Unsharp Masking is a basic image sharpening method and to write this method we need to get help from *gaussianSmoothing* method. After blurring the image, we are taking the difference between the original image and the *blur* to find the details of the image. And later, by adding the details we found to the original image, we are getting a sharpened version of the image.

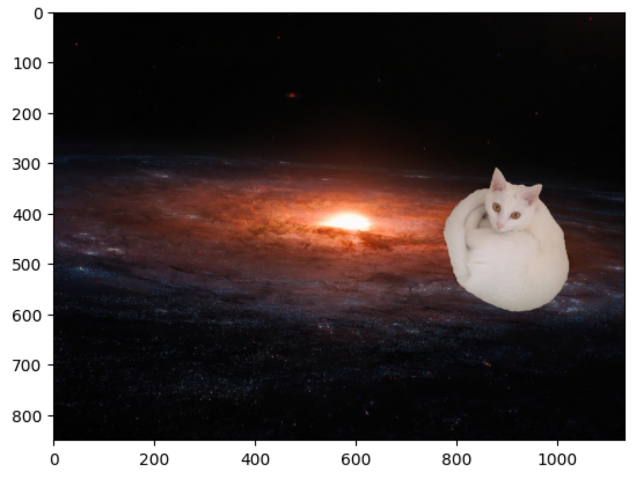
```
def unsharpMasking(img, sigma):
  blur = gaussianSmoothing(img, sigma)
  subtracted = cv2.subtract(img, blur)
  img = cv2.add(subtracted,img)
  return img
```

Testing

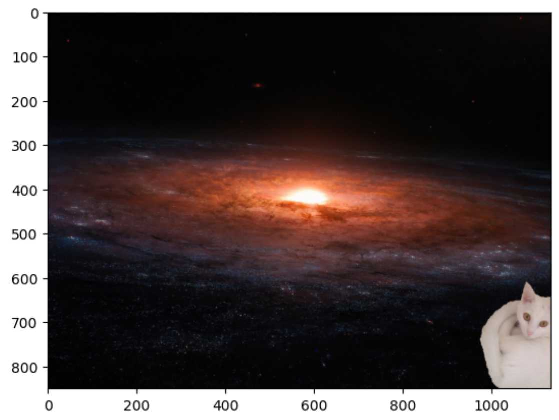
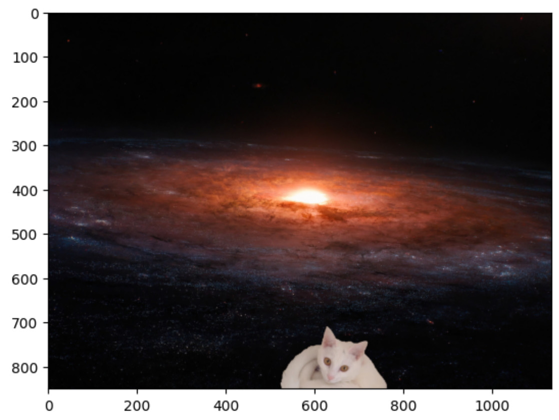
The original images that I have chosen are as follows:



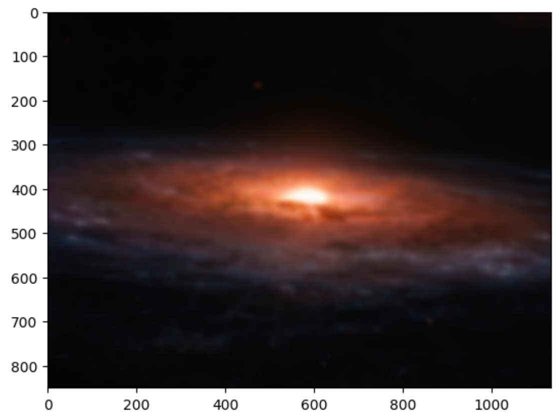
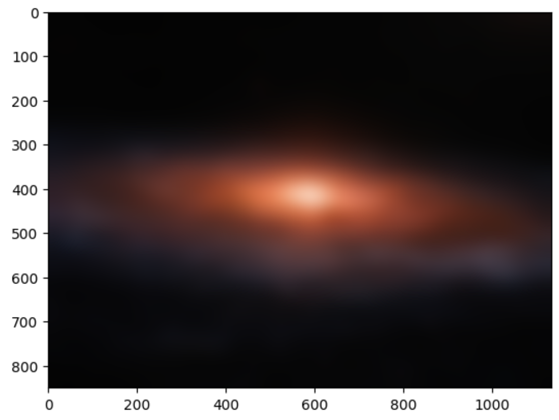
After applying *combineForegroundBackground* method:



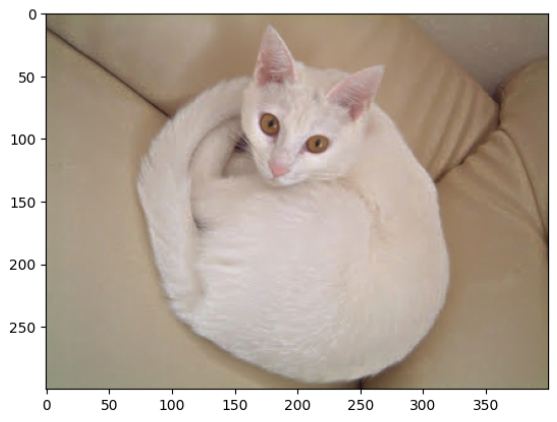
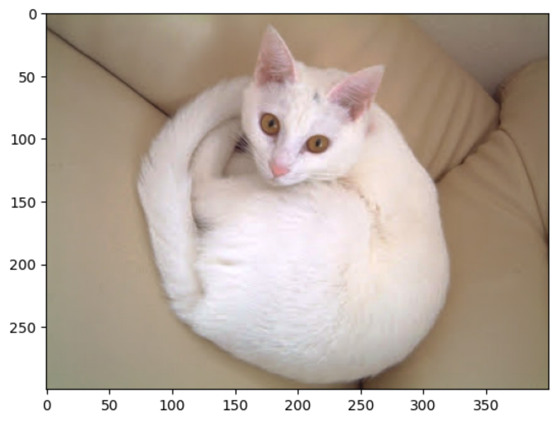
Checking for the boundaries:

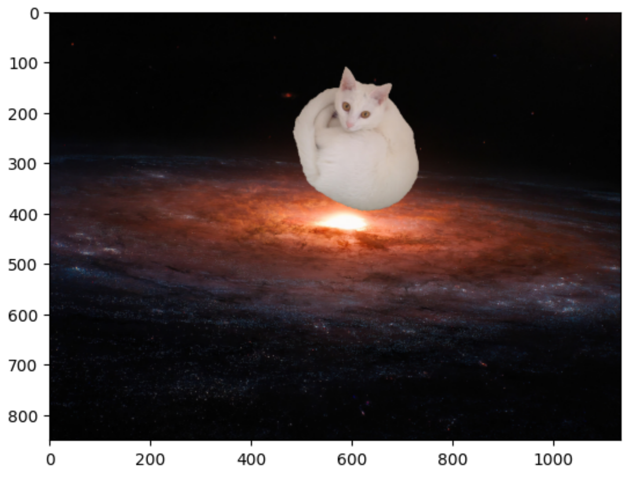
After applying *gaussianSmoothing* to the background with sigma values 5 (left) and 20 (right):

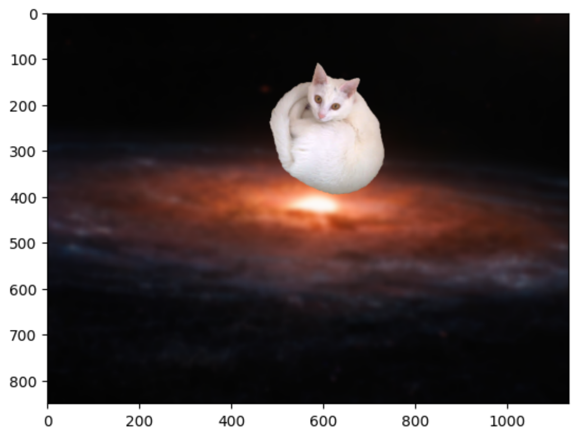
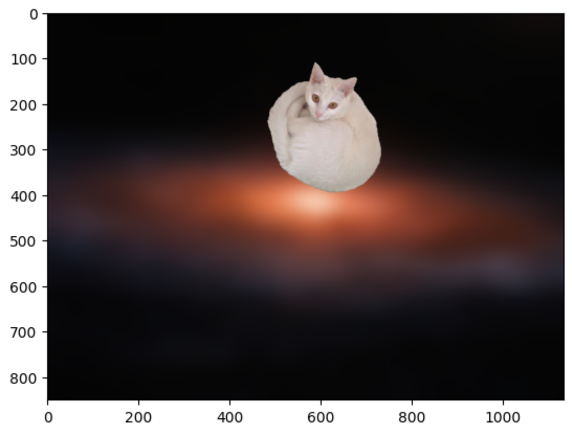
After applying *unsharpMasking* to the background with sigma values 7 (left) and 100 (right):

Combination of foreground and background image without sharpening foreground and smoothing background:



Combination of foreground and background image with sharpening foreground and smoothing background for two different combination of sigma values:

gaussianSmoothing with sigma value 5 & *unsharpMasking* with sigma value 100 (left)

gaussianSmoothing with sigma value 20 & *unsharpMasking* with sigma value 7 (right)