2-gaussian-and-laplacian-pyramids

November 28, 2024

##**Lab Exercise 2: Gaussian and Laplacian Pyramids** • **Objective:** Learn how to build multi-resolution pyramids for an image.

• **Task:** Create Gaussian and Laplacian pyramids for a given image, then use these pyramids to perform image blending between two images.

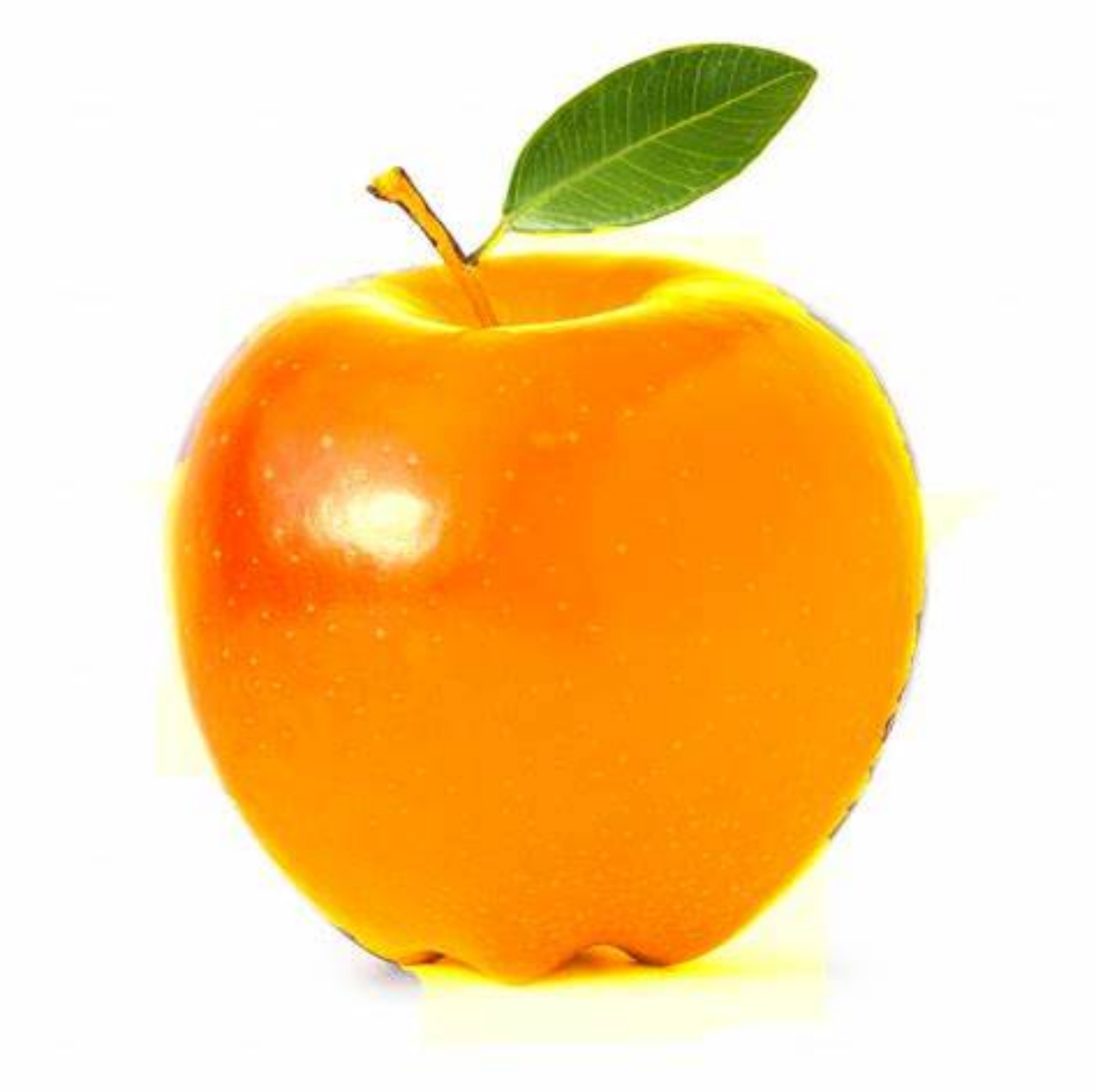
[1]:
```
import cv2 as cv
import numpy as np,sys
from google.colab.patches import cv2_imshow # Import the cv2_imshow function

A = cv.imread('/content/apple.jpg')
B = cv.imread('/content/Orange.jpg')
assert A is not None, "file could not be read, check with os.path.exists()"
assert B is not None, "file could not be read, check with os.path.exists()"
```

[2]:
```
A.shape,B.shape
```

[2]:
```
((612, 612, 3), (473, 474, 3))
```

[3]:
```
cv2_imshow(A)
cv2_imshow(B)
```



[4]:
```
# Define the new dimensions (width, height)
new_width = 400
new_height = 400

# Resize the image
A_resized = cv.resize(A, (new_width, new_height))
B_resized = cv.resize(B, (new_width, new_height))
```

[5]:
```
# Save the resized image

cv.imwrite('Resized_Apple.jpg', A_resized)
cv.imwrite('Resized_Orange.jpg', B_resized)
```

```
image_A = cv.imread('Resized_Apple.jpg')
image_B = cv.imread('Resized_Orange.jpg')
```

[6]:
```
image_A.shape,image_B.shape
```

[6]:
```
((400, 400, 3), (400, 400, 3))
```

[7]:
```
# generate Gaussian pyramid for A
G = image_A.copy()
gpA = [G]
for i in range(6):
    G = cv.pyrDown(G)
    gpA.append(G)

# generate Gaussian pyramid for B
G = image_B.copy()
gpB = [G]
for i in range(6):
    G = cv.pyrDown(G)
    gpB.append(G)
```

[8]:
```
# generate Laplacian Pyramid for A
lpA = [gpA[5]]
for i in range(5,0,-1):
    GE = cv.pyrUp(gpA[i])
    # Resize GE to match the shape of gpA[i-1]
    GE = cv.resize(GE, (gpA[i-1].shape[1], gpA[i-1].shape[0]))
    L = cv.subtract(gpA[i-1],GE)
    lpA.append(L)

# generate Laplacian Pyramid for B
lpB = [gpB[5]]
for i in range(5,0,-1):
    GE = cv.pyrUp(gpB[i])
    # Resize GE to match the shape of gpB[i-1]
    GE = cv.resize(GE, (gpB[i-1].shape[1], gpB[i-1].shape[0]))
    L = cv.subtract(gpB[i-1],GE)
    lpB.append(L)
```

[9]:
```
# Now add left and right halves of images in each level
LS = []
for la,lb in zip(lpA,lpB):
    rows,cols,dpt = la.shape
    ls = np.hstack((la[:,0:cols//2], lb[:,cols//2:]))
    LS.append(ls)
```

[10]:
```
# now reconstruct
ls_ = LS[0]
for i in range(1,6):
    ls_ = cv.pyrUp(ls_)
    # Resize ls_ to match the shape of LS[i] before adding
    ls_ = cv.resize(ls_, (LS[i].shape[1], LS[i].shape[0]))
    ls_ = cv.add(ls_, LS[i])
```

[11]:
```
# image with direct connecting each half
real = np.hstack((image_A[:,:cols//2],image_B[:,cols//2:]))

cv.imwrite('Pyramid_blending2.jpg',ls_)
cv.imwrite('Direct_blending.jpg',real)
```

[11]:
```
True
```

[12]:
```
# reading the images
Direct_Blending = cv.imread('Direct_blending.jpg')
Pyramid_Blending= cv.imread('Pyramid_blending.jpg')
```

[14]:
```
# image with direct connecting each half
real = np.hstack((image_A[:,:cols//2],image_B[:,cols//2:]))

# Ensure the path is correct and the file is created successfully
cv.imwrite('/content/Pyramid_blending2.jpg',ls_)
cv.imwrite('/content/Direct_blending.jpg',real)

# reading the images
# Update the paths to match the previous write operations
Direct_Blending = cv.imread('/content/Direct_blending.jpg')
Pyramid_Blending= cv.imread('/content/Pyramid_blending2.jpg')

cv2_imshow(Direct_Blending)
cv2_imshow(Pyramid_Blending)
```

[ ]: