import cv2 as cv

Normalmente trabajábamos con una imagen de tamaño constante. Pero en algunas ocasiones necesitamos trabajar con (las mismas) imágenes en diferente resolución. Por ejemplo, mientras buscamos algo en una imagen, como una cara, no estamos seguros de qué tamaño estará presente el objeto en dicha imagen. En ese caso, necesitaremos crear un conjunto de la misma imagen con diferentes resoluciones y buscar objetos en todas ellas. Este conjunto de imágenes con diferentes resoluciones se denominan pirámides de imágenes (porque cuando se mantienen en una pila con la imagen de mayor resolución en la parte inferior y la imagen de menor resolución en la parte superior, parece una pirámide).

Hay dos tipos de pirámides de imágenes. 1) Pirámide Gaussiana y 2) Pirámides Laplacianas

El nivel superior (baja resolución) en una pirámide gaussiana se forma eliminando filas y columnas consecutivas en una imagen de nivel inferior (mayor resolución). Luego, cada píxel en el nivel superior está formado por la contribución de 5 píxeles en el nivel subyacente con pesos gaussianos. Al hacerlo, unMETRO× nortela imagen se convierteMETRO/ 2×norte/ 2imagen. Entonces el área se reduce a un cuarto del área original. Se llama Octava. El mismo patrón continúa a medida que ascendemos en la pirámide (es decir, la resolución disminuye). De manera similar, mientras se expande, el área se multiplica por 4 en cada nivel. Podemos encontrar pirámides gaussianas usando las funciones cv.pyrDown() y cv.pyrUp() .

img = cv.imread('resources/messi5.jpg')  
assert img is not None, "file could not be read, check with os.path.exists()"  
lower\_reso = cv.pyrDown(higher\_reso)

Ahora puedes bajar por la pirámide de imágenes con la función cv.pyrUp() .

higher\_reso2 = cv.pyrUp(lower\_reso)

Mezcla de imágenes usando pirámides.

Una aplicación de Pyramids es Image Blending. Por ejemplo, al unir imágenes, necesitará apilar dos imágenes juntas, pero es posible que no se vea bien debido a las discontinuidades entre las imágenes. En ese caso, la combinación de imágenes con Pyramids le brinda una combinación perfecta sin dejar muchos datos en las imágenes. Un ejemplo clásico de esto es la mezcla de dos frutas, naranja y manzana. Vea el resultado ahora para entender lo que estoy diciendo:

import cv2 as cv  
import numpy as np,sys  
A = cv.imread('resources/apple.jpg')  
B = cv.imread('resources/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()"  
# generate Gaussian pyramid for A  
G = A.copy()  
gpA = [G]  
for i in range(6):  
 G = cv.pyrDown(G)  
 gpA.append(G)  
# generate Gaussian pyramid for B  
G = B.copy()  
gpB = [G]  
for i in range(6):  
 G = cv.pyrDown(G)  
 gpB.append(G)  
# generate Laplacian Pyramid for A  
lpA = [gpA[5]]  
for i in range(5,0,-1):  
 GE = cv.pyrUp(gpA[i])  
 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])  
 L = cv.subtract(gpB[i-1],GE)  
 lpB.append(L)  
# 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)  
# now reconstruct  
ls\_ = LS[0]  
for i in range(1,6):  
 ls\_ = cv.pyrUp(ls\_)  
 ls\_ = cv.add(ls\_, LS[i])  
# image with direct connecting each half  
real = np.hstack((A[:,:cols//2],B[:,cols//2:]))  
cv.imwrite('resources/Pyramid\_blending2.jpg',ls\_)  
cv.imwrite('resources/Direct\_blending.jpg',real)

True