**Blending - Solution**

**1.Method Description：**

I used the Image Stack to generate the multiresolution blending at first time, but due to my poor implementation, I wasn’t able to get satisfying results. So I changed to the Image Pyramids covered in class. I first generated Gaussian pyramids for both **img1** and **img2** using the **cv2.pyrDown** function from the OpenCV library. Next I continue to generate the Laplacian Pyramids for both images using the difference between the Gaussian pyramid at one level and the Gaussian pyramid at the next level. Then the Laplacian pyramids for both images are then blended together by concatenating the left half and right half of each level. Eventually, I used **cv2.py2up** function to upsample each level of the pyramids and blend them into one picture.

**2.Oraple:**

**桌子上的苹果

低可信度描述已自动生成**

**3.Blends of my own choice**

At first I tried the blending with the game concept art of my favorite video game, Red Dead Redemption, just to see if my blending algorithm works for other images that can be blended via the vertical seam, even thought the two pictures are not as well-blended as the orange and apple images, I got relatively good results. 人的照片上写着字

描述已自动生成

I did not made any modifications for this, except that I realized two images might have different sizes, so I modified my code so that one image always have the same size with another.

To make the pictures can be blended through the horizontal lines, I modified the code when Laplacian pyramids of two images, instead of using the left and right half of the images, I used upper and lower half of the images. The following is what I got for Oraple:

图片包含 台球, 游戏机, 碗, 苹果

描述已自动生成

To test the horizontal blending, I used two screenshots from video game MEDIUM, here are the results:

人走在街道上

描述已自动生成

图片包含 户外, 建筑, 街道, 城市

描述已自动生成

模糊的夜景

中度可信度描述已自动生成

The two images are extremely similar by themselves, which leads to a perfectly blended image.