

Hybrid image illusion using convolutional filtering

Introduction

Human eye's perception depends strongly on the distance between the object and the observer. If the distance was small, the eye start focusing more on the sharp details of the image (high frequency), however at the far away distances, the eye is only capable of detecting sooth variations (low frequency). In this project we are trying to use that phenomena to create an illusion by making hybrid image, which gets its low frequency content from an image and its high frequency content from another one.

Implementation of the convolutional filter

This function is used to perform convolution operation over RGB and gray scale image . It has two types of padding (zeros and mirror). It takes only odd shaped kernel, and returns the output in the form of an image with the same size as the input image

```
1 def my_imfilter(image, kernel, mode = 'zeros'):  
2     # get the kernel dimesnsions in kh and kw  
3     kh, kw = kernel.shape  
4     # get the hight and width only as the third dimension may exist or  
5     # not  
6     ih = image.shape[0]  
7     iw = image.shape[1]  
8     # get the number of dimesnsions for both the kernel and the image  
9     kdim = kernel.ndim  
10    idim = image.ndim  
11  
12    assert kdim == 2, "kernel dimensions must be exactly two"  
13    assert idim == 2 or idim == 3, "image dimensions must be exactly  
14    two or three"  
15    assert (kh%2) != 0 and (kw%2) != 0, "all kernel dimensions must be  
16    odd"  
17    # this foarmula calculates how many rows do i need to put so that i  
18    # can make proper padding  
19    hpad = (kh-1)//2  
20    wpad = (kw-1)//2  
21  
22    filtered_image = np.zeros_like(image)  
23  
24    if mode == 'zeros':  
25        md = 'constant'  
26    elif mode == 'reflect':  
27        md = 'reflect'  
28    else:  
29        raise Exception('the mode {} is not defined \n "zeros" and "  
30        reflect are available"'.format(x))  
31    # change the padding function according to the input image  
32    if idim == 2:  
33        padded_img = np.pad(image, [(hpad, hpad), (wpad, wpad)], mode=md)
```

```
29     else :
30         paddedImg=np.pad(image,[(hpad,hpad),(wpad,wpad),(0,0)],mode='
31         constant')
32         dim_No=image.shape[2]
33         # apply convolution over the number of channels
34         for dim in range(0,dim_No):
35             for i in range(0,ih):
36                 for j in range(0,iw):
37                     # multiply the kernel with the cropped part of the image and
38                     # then sum the result
39                     cropped=paddedImg[i:i+kh,j:j+kw,dim]
40                     filtered_image[i,j,dim]=np.sum(np.multiply(kernel,cropped))
41     return filtered_image
```

Filter tests and results

Identity filter

Figure 1: *Left:* My result was spectacular. *Right:* Curious.