# 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

```
def my_imfilter(image, kernel,mode ='zeros'):
  # get the kernel dimesnsions in kh and kw
  kh,kw=kernel.shape
  # get the hight and width only as the third dimension may exist or
    not
  ih=image.shape[0]
  iw=image.shape[1]
  # get the number of dimsensions for both the kernel and the image
  kdim=kernel.ndim
  idim=image.ndim

  assert kdim == 2 , "kernel dimensions must be exaxctly two"
  assert idim == 2 or idim ==3 , "image dimensions must be exaxctly
    two or three"
  assert (kh%2) !=0 and (kw%2) !=0 , "all kernel dimensions must be
    odd"
  # this foarmula calculates how many rows do i need to put so that i
    can make proper padding
  hpad=(kh-1)//2
  wpad=(kw-1)//2

  filtered_image = np.zeros_like(image)

  if mode=='zeros':
    md='constant'
  elif mode=='reflect':
    md='reflect'
  else:
    raise   Exception('the mode {} is not defined \n "zeros" and "
    reflect are available"'.format(x))
  # change the padding function according to the input image
  if idim==2:
    paddedImg=np.pad(image,[(hpad,hpad),(wpad,wpad)],mode=md)
```

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

## A Result

1. Result 1 was a total failure, because...
2. Result 2 (Figure 1, left) was surprising, because...
3. Result 3 (Figure 1, right) blew my socks off, because...

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

My results are summarized in Table 1.

| Condition | Time (seconds) |
|---|---|
| Test 1 | 1 |
| Test 2 | 1000 |

Table 1: Stunning revelation about the efficiency of my code.