Project 1 Writeup

Instructions

- Provide an overview about how your project functions.
- Describe any interesting decisions you made to write your algorithm.
- Show and discuss the results of your algorithm.
- Feel free to include code snippets, images, and equations.
- List any extra credit implementation and result (optional).
- Use as many pages as you need, but err on the short side.
- **Please make this document anonymous.**

Project Overview

This project aims to implement two functions `myfilter()` and `gen_hybrid_image()`. Different types of filters are applied to images to test `myfilter()` function. As for `gen_hybrid_image()`, besides testing with extra pairs of images, the effect of image assignment sequence to image1 and image2 is also investigated.

Implementation Detail

Here is the implemetation for `myfilter()`. At first, shapes of the kernel and image are obtained and an error message is raised if the filter has an even-dimension. In order to support both grayscale and RGB pictures, the grayscale picture is reshaped to a 3d array, so that we don't need to worry about the dimensions difference between grayscale and RGB picture. Then, the image is padded with 0 on the hight and width dimension using `np.pad()`. The kernel is flipped using `np.flit()` as we want to realize convolution operation. After all above manipulations, the output is obtained by taken dot product between `flipped_kernel` and `padded_image`. If the input image is grayscale, the output is reshaped back to 2D to keep the result the same dimension as the input.

```
(k, l) = kernel.shape
(m, n, c) = image.shape
if (k * l) % 2 == 0:
    raise Exception("Output with even filters are not defined!")

Grayscale = False
if len(image.shape) == 2:
    Grayscale = True
    image = np.reshape(image, (image.shape[0], image.shape[1], 1))

padded_image = np.pad(image, ((k // 2, k // 2), (l // 2, l // 2), (0,
                                   0)), "constant")
# because we want to calculate convolution, we need to flip the kernel
flipped_kernel = np.flip(kernel)
output = np.zeros(image.shape)
for i in range(m):
    for j in range(n):
        output[i, j] = np.tensordot(flipped_kernel, padded_image[i : i
                                       + k, j : j + l], axes=[(0,
                                       1), (0, 1)])

if Grayscale:
    output = output.reshape(output, (m, n))
filtered_image = output
```

Result

1. Results for different filters

2. results for hybrid with different sequence.

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

Extra Credit (Optional)

1. Pad with reflected image content

```
        one = 1;
        two = one + one;
        if two == 2
            disp( 'This computer is not broken.' );
        end
```

2. own hybrid image

```
one = 1;
two = one + one;
if two == 2
    disp( 'This computer is not broken.' );
end
```

3. FFT-based convolution

```
one = 1;
two = one + one;
if two == 2
    disp( 'This computer is not broken.' );
end
```