Homework 2 Writeup

Instructions

- 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.
- Use as many pages as you need, but err on the short side If you feel you only need to write a short amount to meet the brief, th
- **Please make this document anonymous.**

Algorithm Implementation

In the filtering of an image using convolution, a filter called the kernel is applied to each pixel of the image. The given my_imfilter takes the image and the filter as parameters and returns the ouput. After studying the convolution filtering algorithm, I learned that the filter should be flipped both horizontally and vertically before applying. The flipping of the kernel is done in line 1 of the code below. When applying the kernel on elements at the edge, there should be padding around it so that the size of the kernel and the size of the element and its neighbor match. The width of the padding is half the size of the kernel at respective dimensions. The calculation of the padding size and padding the image is done in lines 2 and 3. After applying filter to the padded image, the resulting matrix is the same size as the padded image. Thus, a zero matrix of the same size as padded image is created in line 4. The given image is in RGB space, so the for loop is iterated through the third dimension of the image matrix in line 5. Only the pixels that are not paddings are iterated in the double for loop in lines 6 and 7. The target area centered at the pixel (i,j) is accessed in line 8. The corresponding element in the resulting filtered image is updated by the sum of elements of the element-wise multiplication between the target matrix and the kernel. This is the main idea of convolution filtering and shown in line 9. After the triple for loops, the filtered image should be cropped back to match the original image size. The crop is done in line 13.

```
1 flipped_kernel = flip(flip(filter, 1), 2);
2 pad_size = floor(size(filter)/2);
3 padded_image = padarray(image, [pad_size, 0]);
4 filtered_image = zeros(size(padded_image));
5 for d=1:size(padded_image, 3)
6     for i=pad_size(1)+1:size(image, 1)+pad_size(1)
7         for j=pad_size(2)+1:size(image, 2)+pad_size(2)
8             target = padded_image(i-pad_size(1):i+
                  pad_size(1), j-pad_size(2):j+pad_size(2),
                  d);
```

```
            filtered_image(i, j, d) = sum(sum(target.*
               flipped_kernel));
        end
    end
end
output=filtered_image(pad_size(1)+1:size(image,1)+
   pad_size(1), pad_size(2)+1:size(image,2)+pad_size(2),
   :);
```

Attempts

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

My code snippet highlights an interesting point.

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

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...

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.

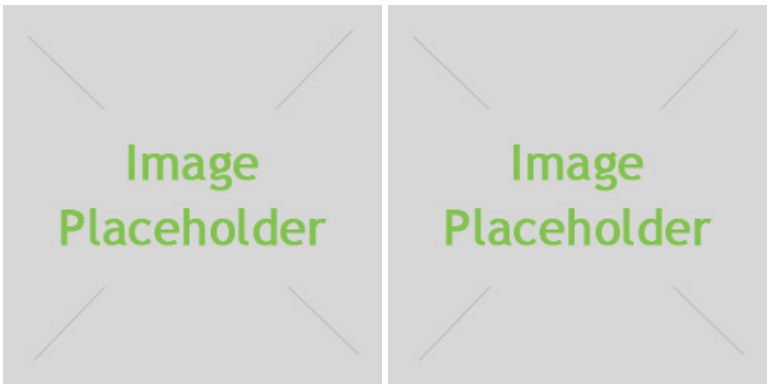

Figure 1: *Left:* My result was spectacular. *Right:* Curious.