

## Assignment 6

**def reduce(image):**

Like many of the following functions, just worked through the instructions.

```
kernel = generatingKernel(0.4);
reduced = scipy.signal.convolve2d(image, kernel, 'same');
return reduced[::2, ::2];
```

The first step was taken directly from the first line of the instructions: generating a kernel of a parameter of 0.4. Then convolve with the function in the readme. Then reduce by 2, using every other space. I went to <https://docs.scipy.org/doc/numpy/reference/arrays.indexing.html> to learn indexing and found that ::2 means to use every other item.

This entire function could have been decreased to:

```
return scipy.signal.convolve2d(image, generatingKernel(0.4), 'same')[::2, ::2];
```

But I like seeing it one step at a time for reading purposes.

---

**def expand(image):**

Very similar to the reduce function, just the other way.

First you generate the kernel like before, but this time you should create a new image matrix that's double both the height and width as we are doubling the size. On the Array Indexing page, I found you can set the indexes the same way you do the get. Once you do the convolve, you must multiple the result by four.

```
kernel = generatingKernel(0.4);
expanded = np.zeros((image.shape[0] * 2, image.shape[1] * 2));
expanded[::2, ::2] = image;
expanded = scipy.signal.convolve2d(expanded, kernel, 'same');
return expanded * 4;
```

You must multiply it by four because you double the width and the height. Once it convolves with the kernel, it places values where the images didn't index into the expanded image. So this image is darker as it spread the values within the surrounding pixels at  $1/(2 \text{ for the width} * 2 \text{ for the height})$  which is closer to 0 (which is black) than 255 (which is white).

---

```
def gaussPyramid(image, levels):
```

Looking at the needed output, you need to return an array with output[0] being the image. Then there needs to be output[n] = image reduced that number of times. Therefore, just did a quick iteration appending the reduced previously appended array.

```
    output = [image]
    for level in range(levels):
        output.append(reduce(output[level]));
    return output
```

---

```
def laplPyramid(gaussPyr):
```

The pyramid is the same size as the Gaussian pyramid; therefore, I iterated over the size of the previous function. I used the function in the instructions: output[k] = gauss\_pyr[k] - expand(gauss\_pyr[k + 1]) to append to the output. Since there is no length + 1, the last element will be the last element of the Gaussian Pyramid. With the note, I needed to crop the image to do a matrices subtraction, using the index splicing from before.

```
    output = [];
    for index in range(len(gaussPyr) - 1):
        shape = gaussPyr[index].shape;
        output.append(gaussPyr[index] - expand(gaussPyr[index + 1])[:shape[0],
            :shape[1]]);
    output.append(gaussPyr[-1]);
    return output
```

---

```
def blend(laplPyrWhite, laplPyrBlack, gaussPyrMask):
```

For this function, I did a for loop through each in the Gaussian pyramid mask and then used the function in the instructions: output[i, j] = current\_mask[i, j] \* white\_image[i, j] + (1 - current\_mask[i, j]) \* black\_image[i, j]. I removed the i,j in favor of index.

```
    blended_pyr = []
    for index in range(len(gaussPyrMask)):
        blended_pyr.append(gaussPyrMask[index] * laplPyrWhite[index] + (1 -
            gaussPyrMask[index]) * laplPyrBlack[index]);
    return blended_pyr
```



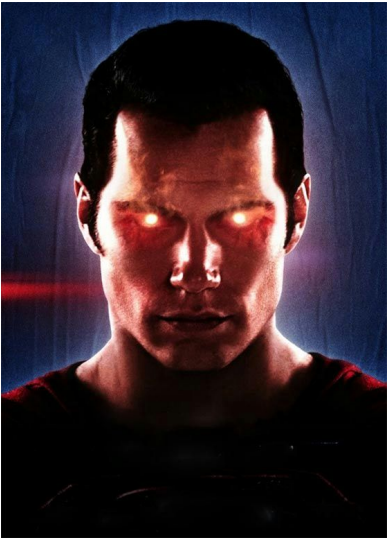
---

```
def collapse(pyramid):
```

For the final function, I start with the smallest layer and then expand it while adding to the smallest image. I continue doing this for the entire array until the items are all added back together.


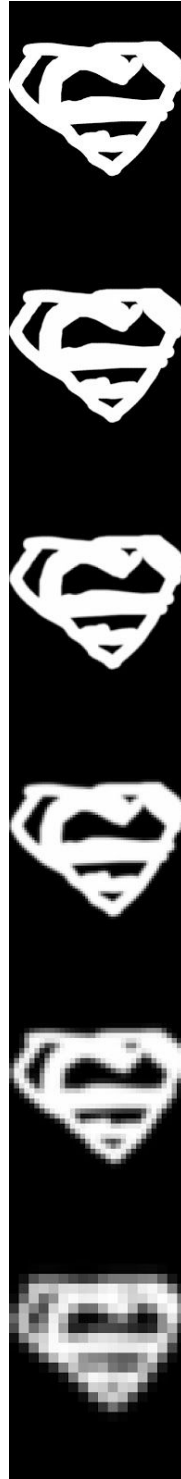

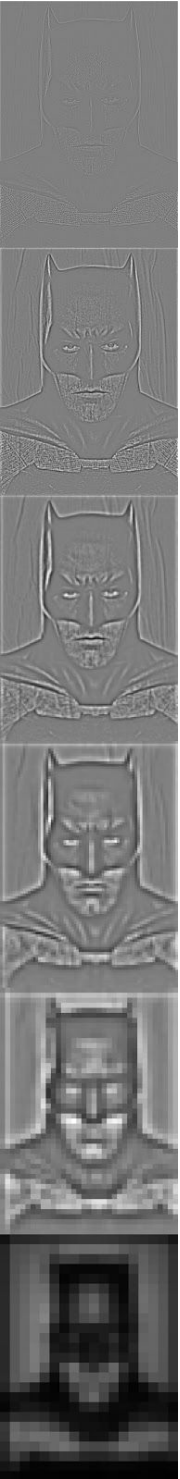

```
output = pyramid[-1]
for index in range(len(pyramid)-1, 0, -1):
    shape = pyramid[index - 1].shape;
    output = pyramid[index - 1] + expand(output)[:shape[0], :shape[1]];
return output
```

**Examples:**

		
<b>Black</b>	<b>Mask</b>	<b>White</b>

**Sources:**

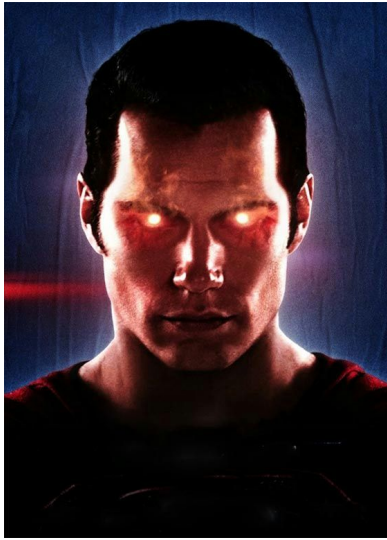
- Batman:  
[http://cdn-static.denofgeek.com/sites/denofgeek/files/styles/insert\\_main\\_wide\\_image/public/7/69//ben\\_affleck\\_batman.jpg?itok=J1tGlk37](http://cdn-static.denofgeek.com/sites/denofgeek/files/styles/insert_main_wide_image/public/7/69//ben_affleck_batman.jpg?itok=J1tGlk37)
- Superman:  
[http://orig14.deviantart.net/5bdc/f/2015/122/7/4/batman\\_v\\_superman\\_dawn\\_of\\_justice\\_poster\\_9\\_by\\_camw1n-d8rq49x.jpg](http://orig14.deviantart.net/5bdc/f/2015/122/7/4/batman_v_superman_dawn_of_justice_poster_9_by_camw1n-d8rq49x.jpg)

					
Gauss Black	Gauss Mask	Gauss White	Lapl Black	Lapl White	Sample Out

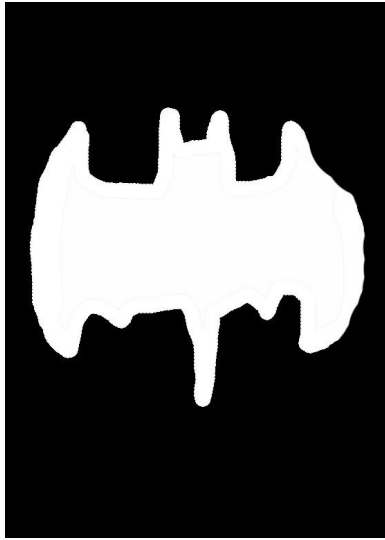


I made the images the same size and then created a mask via photoshop with the paint brush. I thought it was cool Batman / Superman image with the Superman logo first, then Batman.









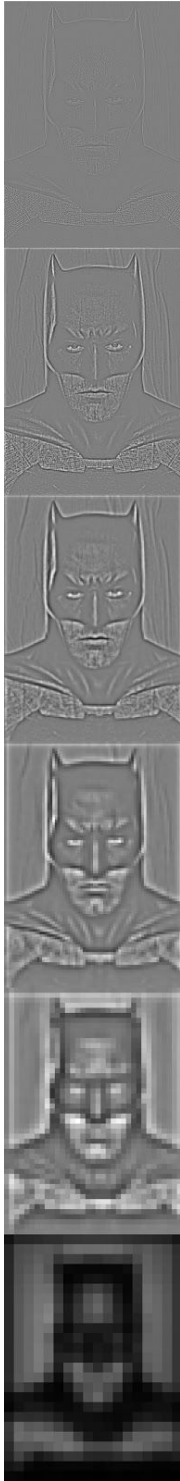
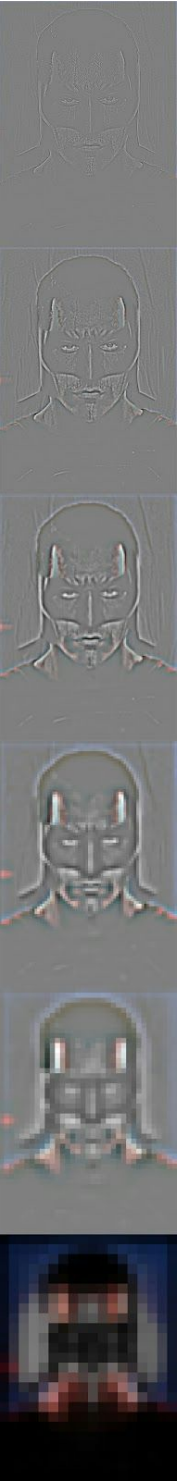
**Black**



**Mask**



**White**

					
Gauss Black	Gauss Mask	Gauss White	Lapl Black	Lapl White	Sample Out

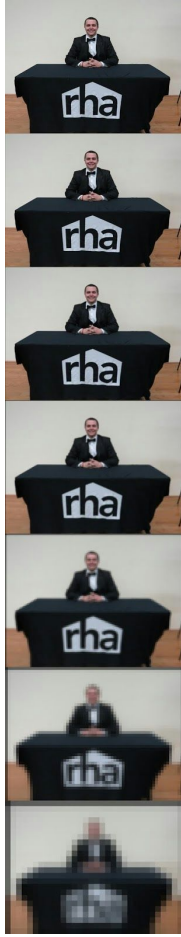
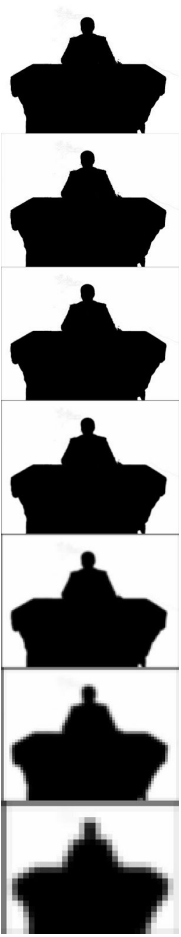
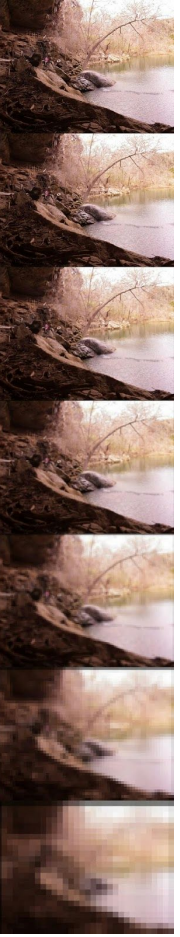

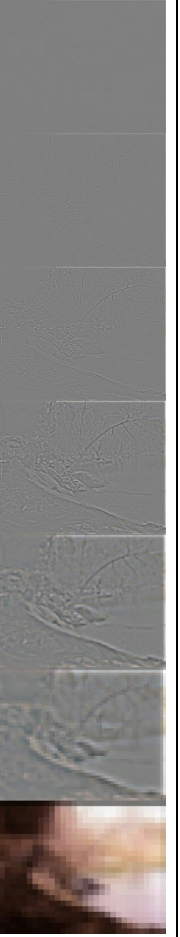





Final Examples, My Images - Putting me in different spots


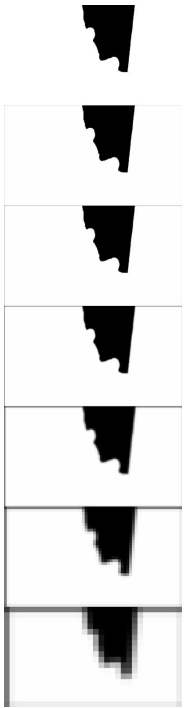


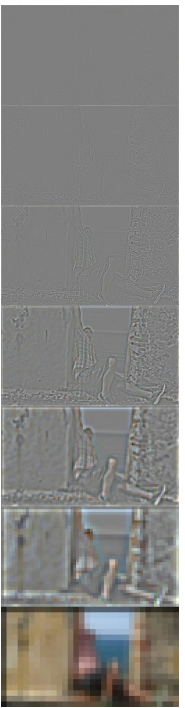
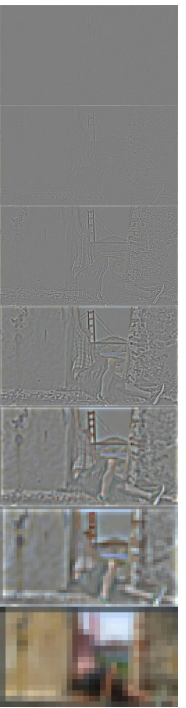




					
Gauss Black	Gauss Mask	Gauss White	Lapl Black	Lapl White	Sample Out

Made sure both images were the same size. Then used Photoshop's wizard selection to mask myself out of the image and then paint bucketed it black with the rest being white.



					
Gauss Black	Gauss Mask	Gauss White	Lapl Black	Lapl White	Sample Out

Made sure both images were the same size by adding whitespace to the black image. Then used Photoshop's wizard selection to mask myself out of the image and then paint bucketed it black with the rest being white. I am in San Cristobal in Puerto Rico, except now with the Golden Gate bridge behind me.