# Project 1 Writeup

## Project Overview

The coding section of my project consists of a filter and a hybrid image generator. For the filter, I make sure the kernel has an odd by odd dimensionality, and if it does, I iterate over the image pixels and multiply the corresponding values of the kernel to the image and sum them to determine the output pixels. For the hybrid image generator, I apply a Gaussian kernel to the first image using my filter function to get the low-pass image, then I subtract the low pass of the second image from the second image itself to get the high-pass image. Then I add the two images together to get the hybrid.

## Implementation Detail

For my implementation of the filter function, I make a padded copy of the image with half the filter's width and height added onto both ends of the image's width and height, respectively. I then iterate over the padded image, modifying the corresponding pixels on the final output image. I handle grayscale and color images separately, since color images have an extra dimension. An interesting decision I made was that I reshaped both my filter and the image section into vectors and then calculated the dot product in order to save on some computation time as opposed to using a for loop.

## Result

My results seem to be exactly as expected.
Figures 1-3 are the results of my filter algorithm on marilyn.jpg
Figures 4-5 are the results of my hybrid algorithm on the default dog and cat

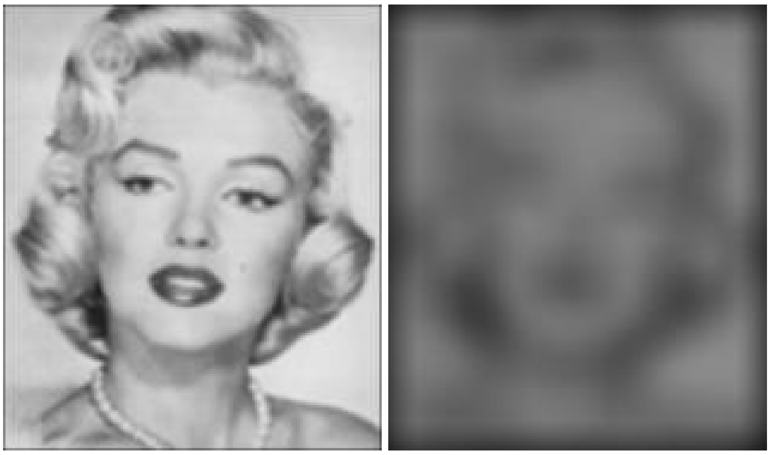

Figure 1: *Left:* blur *Right:* large blur

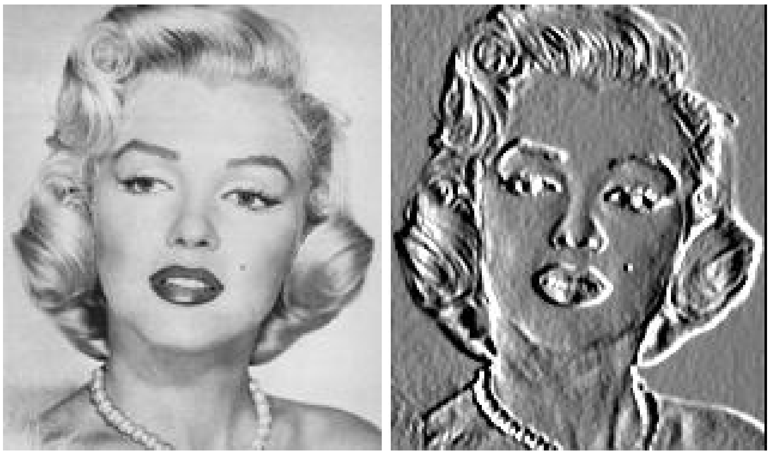

Figure 2: *Left:* identity *Right:* sobel

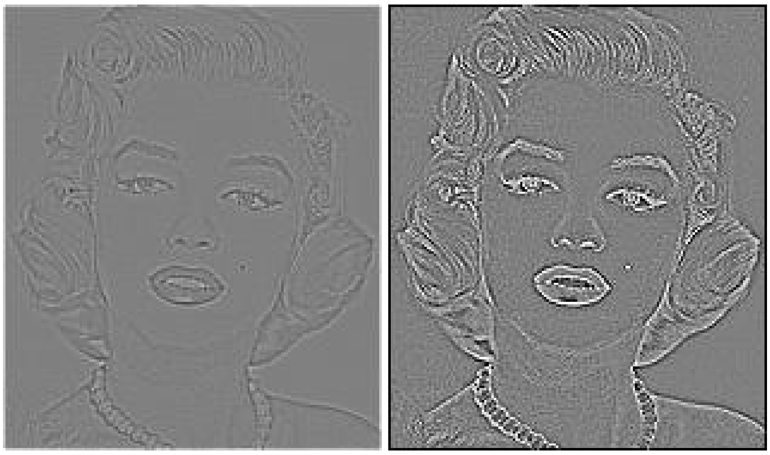

Figure 3: *Left:* high pass *Right:* laplacian

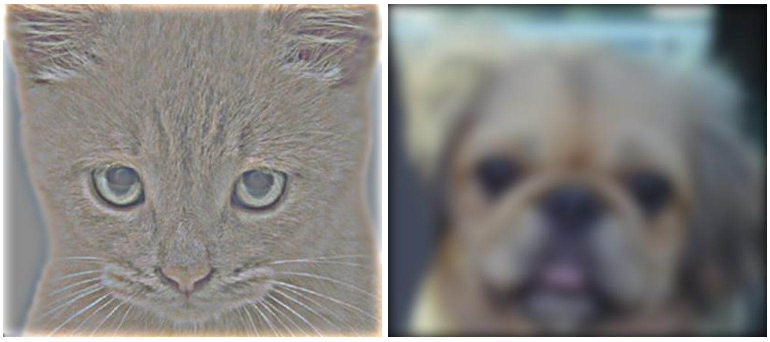

Figure 4: *Left:* high frequencies *Right:* low frequencies

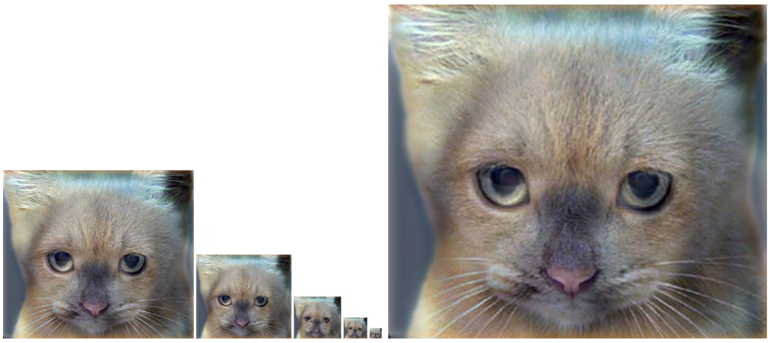

Figure 5: *Left:* hybrid image scales *Right:* hybrid image