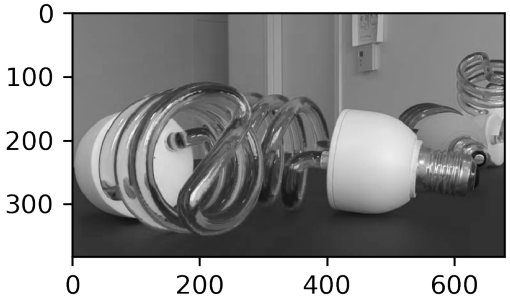
Convolutions, Fourier Transforms, and Image Pyramids*

Benjamin Geyer[1]

I. IMAGE FILTERS

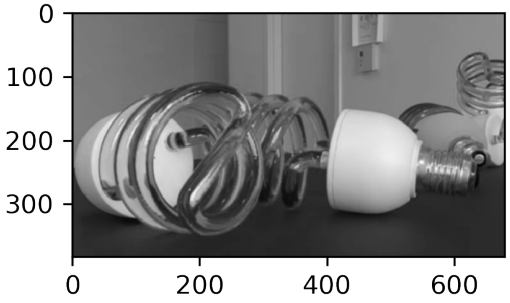
A. Simple Filters

Fig. 1. My original image - Identity Filter.



$$f_a = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Fig. 2. Filter A - Box Filter.



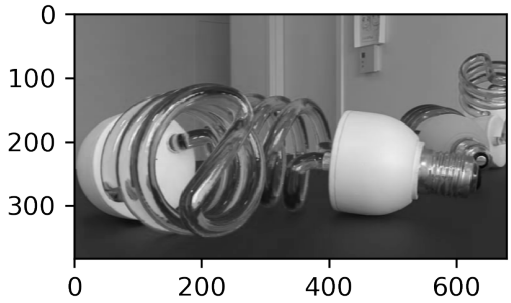
$$f_a = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

The Box Filter is a very basic smoothing filter which averages each 9 pixel block equally. This can be seen as the image is much blurrier and darker than the original image. The filter can be separated into 3 parts: the scalar $\frac{1}{9}$ to average the sum, and the two main 1-D filters of $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ to get each value of the surrounding pixels.

*This work was not supported by any organization

[1]B. Geyer is a Graduate student in the Computer Science Department of the Volgenau School of Engineering, George Mason University, Fairfax, VA. Contact at bgeyer3@gmu.edu

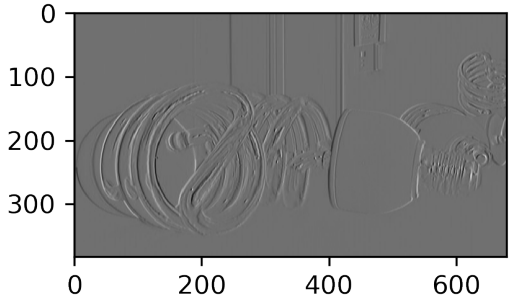
Fig. 3. Filter B - Horizontal Smoothing.



$$f_a = \frac{1}{3}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

The Horizontal Smoothing Filter averages horizontal sets of pixels as can be seen where the image's vertical lines on the door-frame are slightly blurrier and more muddled than the original image. The filter can be separated into 3 parts: the scalar $\frac{1}{3}$ to average the sum, and the two main 1-D filters of $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ to combine and only get pixels on the same row.

Fig. 4. Filter C - Vertical Edge Filter.

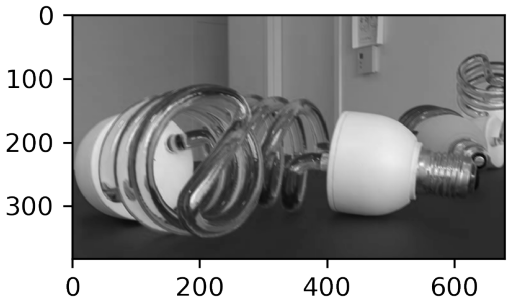


$$f_c = \frac{1}{6}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

This Vertical Edge Filter detects an increase in brightness going from left to right. This can be observed in the image on the coils on the left and the left side of the base of the lightbulb on the right. This could be combined with a flipped Vertical Edge Filter detecting changes in the opposite

direction to find vertical edges on both sides of objects that go from light to dark and then back to light again.

Fig. 5. Filter D - Negative Slope Diagonal Line Filter.



$$f_d = \frac{1}{3}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure D is similar to Figure B in that it performs smoothing along a single axis, only instead of horizontally, it applies it diagonally. This has the effect of blurring most parts of the image, but especially on edges curving diagonally up and right such as on the coils on the left of the image. This is the only filter of the 4 that was not separable.

B. Image Derivatives

Fig. 6. Horizontal Sobel Filter

Fig. 7. Vertical Sobel Filter

Fig. 8. Image Gradient Magnitude

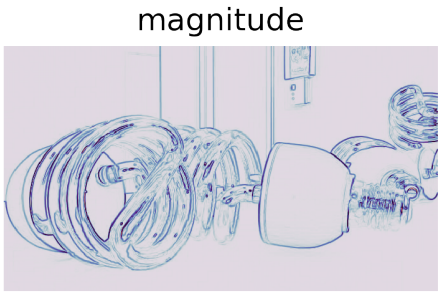


Fig. 9. Image Gradient Direction

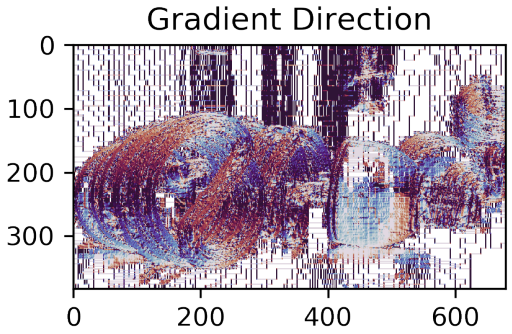


Fig. 10. Image Laplacian

$$laplacian filter = \begin{bmatrix} 0 & 1 & 0 \\ -1 & -4 & 2 \\ 0 & 1 & 0 \end{bmatrix}$$

C. Gaussian Filtering

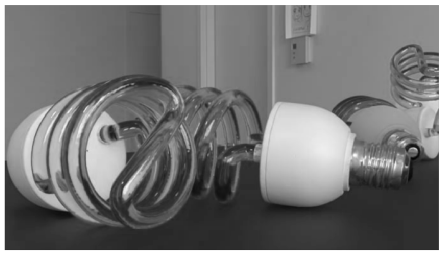
What would happen if the filter width were too small compared to σ?

Having a filter width that is too small for a given σ is an issue because then the filter cuts off some of the tails of the Gaussian. 3 standard deviations in each direction contains most of the distribution of a Gaussian so having a width of at least 3σ allows the filter to sufficiently contain all of the possible values. cutting off too early can reduce the contrast of an image by truncating the values at the extreme.

$$g = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

The σ of g is approximately 0.8.

Fig. 11. $\sigma = 0.5$



$f_{10} =$

$$\begin{bmatrix} 7.16e-08 & 2.89e-05 & 2.13e-04 & 2.89e-05 & 7.16e-08 \\ 2.89e-05 & 1.16e-02 & 8.61e-02 & 1.16e-02 & 2.89e-05 \\ 2.13e-04 & 8.61e-02 & 6.36e-01 & 8.61e-02 & 2.13e-04 \\ 2.89e-05 & 1.16e-02 & 8.61e-02 & 1.16e-02 & 2.89e-05 \\ 7.16e-08 & 2.89e-05 & 2.13e-04 & 2.89e-05 & 7.16e-08 \end{bmatrix}$$

Fig. 12. $\sigma = 0.5$

$$f_{11} = \begin{bmatrix} 0.00291 & 0.01306 & 0.02153 & 0.01306 & 0.00291 \\ 0.01306 & 0.05854 & 0.09653 & 0.05854 & 0.01306 \\ 0.02153 & 0.09653 & 0.15915 & 0.09653 & 0.02153 \\ 0.01306 & 0.05854 & 0.09653 & 0.05854 & 0.01306 \\ 0.00291 & 0.01306 & 0.02153 & 0.01306 & 0.00291 \end{bmatrix}$$

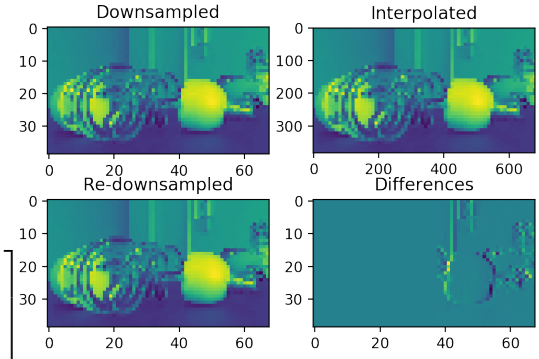
$$f_{12} = \begin{bmatrix} 0.01195 & 0.02328 & 0.02908 & 0.02328 & 0.01195 \\ 0.02328 & 0.04535 & 0.05664 & 0.04535 & 0.02328 \\ 0.02908 & 0.05664 & 0.07073 & 0.05664 & 0.02908 \\ 0.02328 & 0.04535 & 0.05664 & 0.04535 & 0.02328 \\ 0.01195 & 0.02328 & 0.02908 & 0.02328 & 0.01195 \end{bmatrix}$$

D. Derivative of Convolution

Insert Here

Fig. 13. $\sigma = 0.5$

Fig. 14. Nearest Neighbor Upsampling

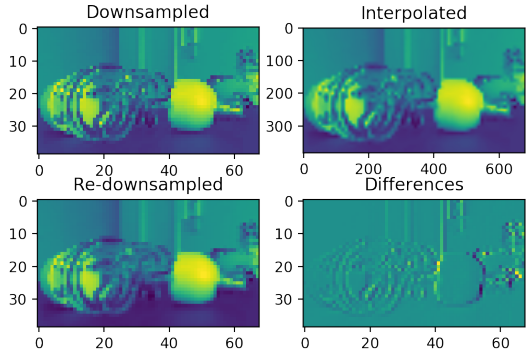


II. IMAGE SUBSAMPLING

A. Upsampling Kernels

Nearest Neighbor Upsampling generates the same quality image but just with more pixels. The re-downsampled image is nearly identical other than an extra pixel column being inserted in the middle for some reason during downsampling, shifting the right half of the image over. This is shown in Fig. 13.d **Differences** which shows slight differences horizontally on the right half of the image.

Fig. 15. Bilinear Upsampling

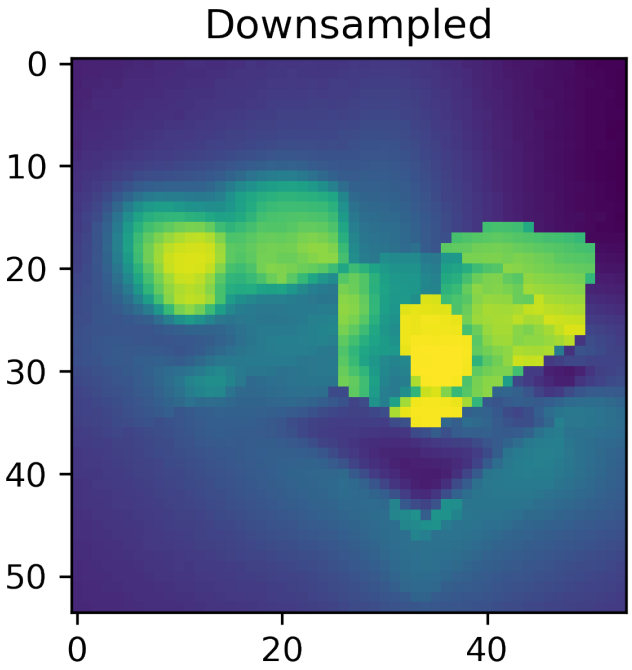


Bilinear Upsampling generates a better quality image that smoothens out jagged edges along edges in images. The re-downsampled image is similar to the original but not exactly the same. This is shown in Fig. 14.d **Differences** which shows differences especially along sharp edges along objects

such as the spirals on the lightbulb on the left.
Unfortunately, I could not finish my implementation for Cubic Interpolation. I Know I need to use finite differences to calculate the derivative at a given point to calculate a spline to connect the two known points. Then, I can estimate new values for pixels based on their approximate location along the spline. This can be done with the following function:

B. Upsampling with Fourier Transforms

Fig. 16. Original Downsampled Image



In Fig. 16 **Frequency-Space Representation**, most of the weight is gathered around the four corners of the image because there are a few low-frequency waves that have very high amplitude and make up most of the signal. This is especially evident in the top left corner where a really bright pixel makes up most of the weight of the entire image.

In Fig. 17 **Scaled Up Frequency-Space Representation**, again, most of the weight is gathered around the four corners of the image. However, this image represents the frequency space in a much higher "resolution" and therefore, the four corners are smoother and more uniform. However, this is not the frequency space we want to use because we want to shift and pad the smaller "resolution" frequency space and then inverse it at a higher "resolution."

Fig. 18 **FFT Shifted Interpolation**, shows the interpolated image from an FFT Shifted frequency space into a image-space upsampled image. The re-downsampled image on the right is exactly the same as the original downsampled image as expected. However, the interpolated image seems to have not properly disposed of some frequencies because there are "echos" of waves farther out from each edge most likely from higher frequency waves that are lower amplitude but weren't filtered out.

Fig. 17. Frequency-Space Representation

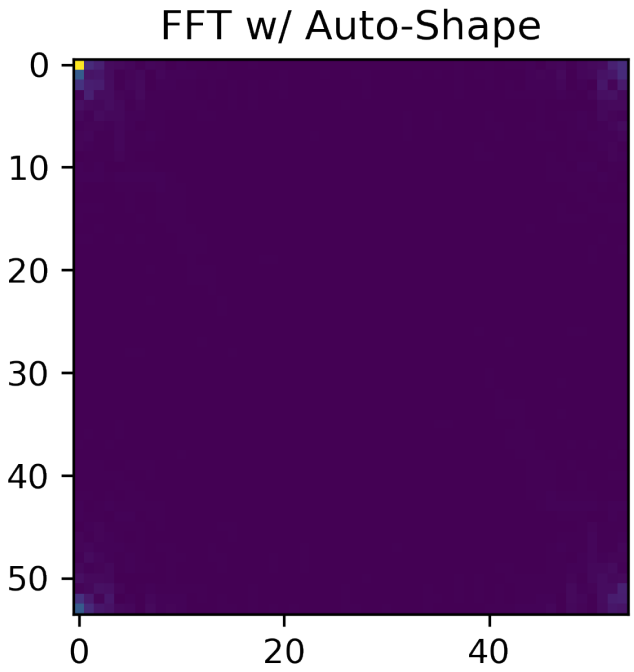
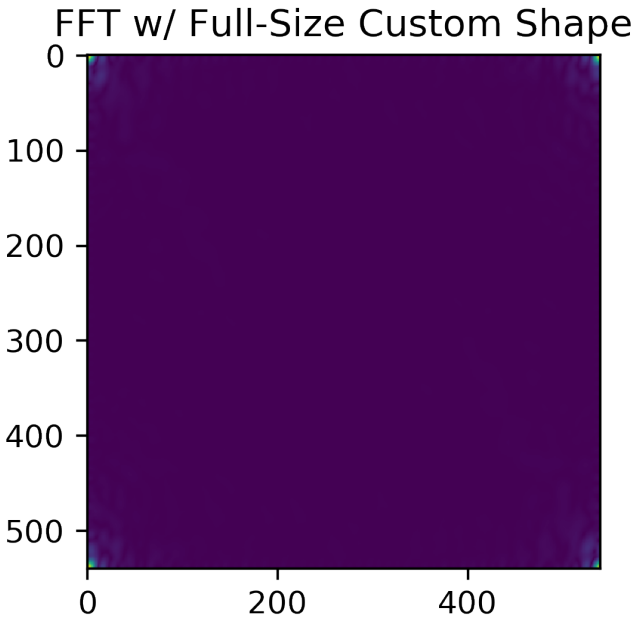


Fig. 18. Scaled Up Frequency-Space Representation



I thought that perhaps I had done the shifting wrong in frequency space so I tried IFFT shifting the image after transforming it into frequency space. Fig. 19 **IFFT Shifted Interpolation**, shows the interpolated image and the re-downsampled image on the right is exactly the same as the original downsampled image as expected. However, the interpolated image again seems to have not properly disposed of some higher frequencies.

In order to attempt to fix the shifting problem, I tried

Fig. 19. FFT Shifted Interpolation

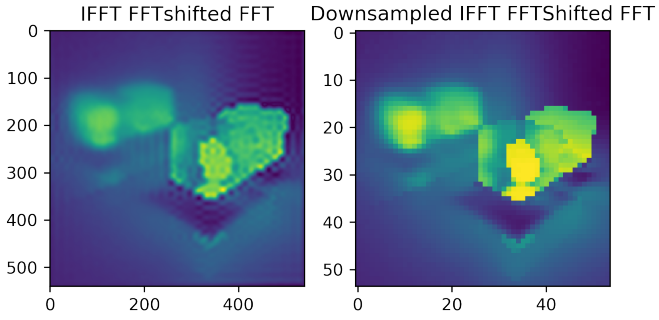


Fig. 20. IFFT Shifted Interpolation

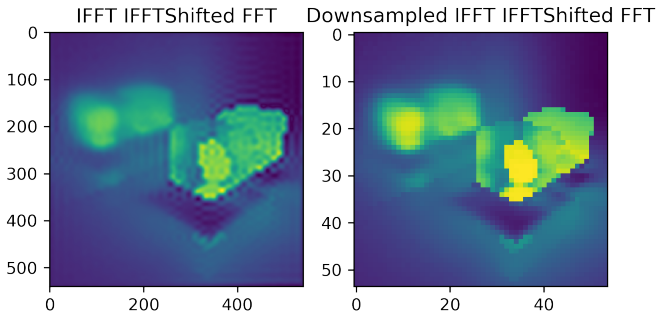
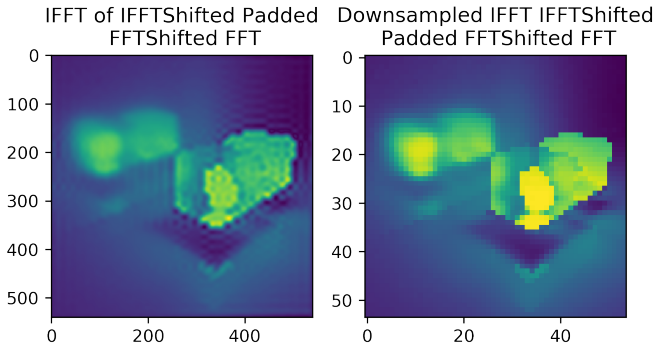


Fig. 21. IFFT Shifted Zero-Padded FFT Shifted Interpolation



FFT shifting, zero-padding the image, and then IFFT shifting back before interpolating the image. Fig. 21 **IFFT Shifted Zero-Padded FFT Shifted Interpolation**, shows the interpolated image once again containing these waves. Rather than creating smooth pixel data in a certain direction, the transform actually introduced patterns similar to Moire patterns resulting from higher frequencies of waves lining up. I believe that I needed to use a threshold to cut off any frequencies below a certain weight / amplitude in order to only select the "correct" waves that actually generated the image. I thought that zero-padding would shift more sampling to the specific areas of the image and result in a smoother result, but this was not evident. Another technique would be to simply select the k waves with the highest amplitude for greater flexibility across datasets. This can prevent aliasing from not obeying the Nyquist limit from occurring where the sampling rate allows higher frequencies to match the sampled data while causing distortions between sampled pixels. This is similar to the Wagon Wheel effect where the speed of a wheel rotating forwards matches the frequency of a lower backwards-rotating wheel, giving the impression that the wheel is slowly spinning backwards instead of forwards. Nonetheless, the downsampled versions of each FFT-Upsampled image are exactly the same as the original downsampled image in the first place.

Fig. 22. Freq-Space Interpolated Images

Fig. 22 **Freq-Space Interpolated Images**, shows the frequency space interpolated image after interpolation. It is very hard to tell any differences at this resolution so I used the mathematical difference between the images shown below.

Fig. 23. Difference in Freq-Space Interpolated Images

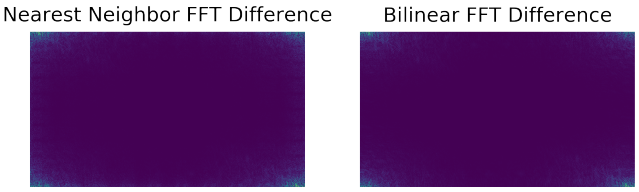


Fig. 23 **Difference in Freq-Space Interpolated Images**, shows the difference in frequency space of the interpolated images compared to the original image. The main differences are at the corners and along lines of similar angles pointing in towards the center. The two are quite similar but bilinear seems slightly better which would be expected from a better interpolation algorithm.

III. HYBRID IMAGES

A. *Originals*

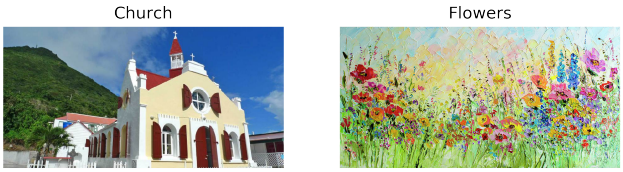
Fig. 24 **Inspiration** is the image that gave me the idea for the images to choose to make a hybrid of. It is a church with a mural on the side. With Hybrid images in mind, I imagined taking a painting and a building and applying the painting to the side of the building.

Fig. 25 **Base Images** shows the two images that I chose to make a hybrid of. My goal was to have the flowers show up on the side of the building if you squint and stand farther away but have the building be the most apparent object in the image.

When I originally started, I didn't realize that most hybrid images don't use wide varieties of color and they usually just

Fig. 24. Inspiration

Fig. 25. Base Images



shift to gray or some similar color scale. However, I found this was an interesting challenge that I could overcome by computing the hybrid of each color and then stacking them back up as a final image which worked wonderfully.

Fig. 26. Church Color Channels

Fig. 26 **Church Color Channels** shows the different color channels present in the church image. As you can see, the different channels are quite slightly different but the objects in the image can be clearly identified in each.

Fig. 27. Flowers Color Channels

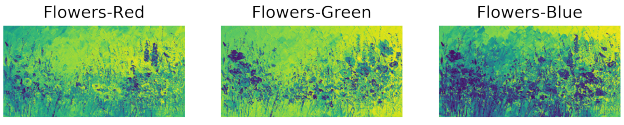


Fig. 27 **Flowers Color Channels** shows the different color channels present in the flowers image. As you can see, the different channels are quite different and show why I had to combine all 3 to get a clear image.

Fig. 28 **Low-Passed Church, High-Passed Flowers - Red** shows the red channel low passed on the church and high passed on the flowers as well as the resulting hybrid.

Fig. 29 **Low-Passed Flowers, High-Passed Church - Red** shows the red channel low passed on the flowers and high passed on the church as well as the resulting hybrid.

Fig. 28. Low-Passed Church, High-Passed Flowers - Red

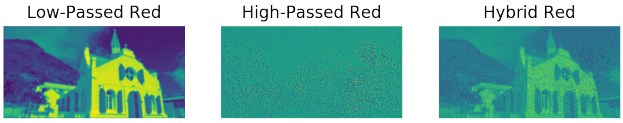


Fig. 29. Low-Passed Flowers, High-Passed Church - Red

Fig. 30. Low-Passed Church, High-Passed Flowers - Green

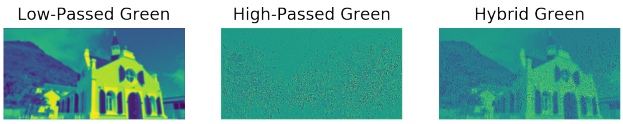


Fig. 30 **Low-Passed Church, High-Passed Flowers - Green** shows the green channel low passed on the church and high passed on the flowers as well as the resulting hybrid.

Fig. 31. Low-Passed Flowers, High-Passed Church - Green

Fig. 31 **Low-Passed Flowers, High-Passed Church - Green** shows the green channel low passed on the flowers and high passed on the church as well as the resulting hybrid.

Fig. 32. Low-Passed Church, High-Passed Flowers - Blue

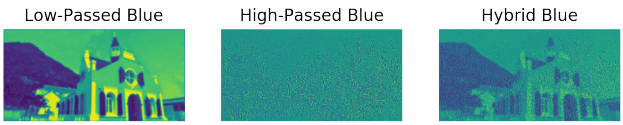


Fig. 32 **Low-Passed Church, High-Passed Flowers - Blue** shows the blue channel low passed on the church and high passed on the flowers as well as the resulting hybrid.

Fig. 33. Low-Passed Flowers, High-Passed Church - Blue

Fig. 33 **Low-Passed Flowers, High-Passed Church - Blue** shows the blue channel low passed on the flowers and high passed on the church as well as the resulting hybrid.

B. Color Hybrids

Stacking the 3 hybrids from each color channel gives the following hybrid images:

Fig. 34. Low-Passed Church, High-Passed Flowers - Color

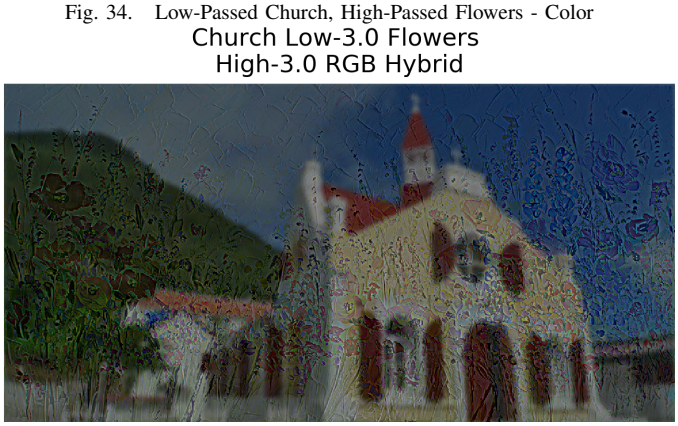


Fig. 34 **Low-Passed Church, High-Passed Flowers - Color** shows all 3 color channels low passed on the church and high passed on the flowers as well as the resulting hybrid.

Fig. 35. Low-Passed Flowers, High-Passed Church - Color

Fig. 35 **Low-Passed Flowers, High-Passed Church - Color** shows all 3 color channels low passed on the flowers and high passed on the church as well as the resulting hybrid.

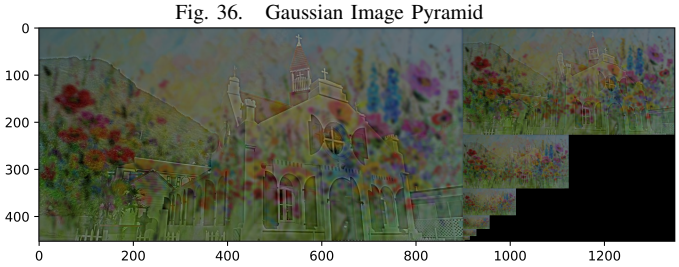
As you can see, Fig. 32 **Low-Passed Flowers, High-Passed Church - Color** does a much better job of capturing the image and getting a clear separation. This is due to the low pass on the flowers picking up the color much more. The high pass loses much more color but that doesn't matter at all for the church. Arguably, the colors of the flower picture make up most of the content with edges following close behind. However, even some of the red of the roof tiles can be observed if the hybrid is closely examined.

All of these hybrids were determined using a σ of 3.0 with a filter size of 11 for both the low-pass and high-pass filters. I actually did test a large range of unique sigmas $[0.5 : 3.5]$ for both filters. However, I found that a simple 3.0 for both captured the most detail and separation for the low-passed flower hybrid image. The low-passed church hybrid images hardly changed due to the colors of the flowers being completely lost in the high-pass filter.

The low-passed flowers performed much better when using individual color channels due to their colors not being washed out, but the images were still quite indistinguishable from each other because the flowers individual color channels were originally very noisy.

C. Pyramid

Fig. 36. Gaussian Image Pyramid



The generated Gaussian Image Pyramid displays how the low pass flowers become more and more dominant as the figure is progressively Gaussian blurred as expected.