Programming Assignment 4

CS 474

`https://github.com/alexander-novo/CS474-PA4`

Alexander Novotny
50% Work
Section 2 (report) and section 3

Matthew Page
50% Work
Section 1 and section 2 (code)

Due: December 9, 2020
Submitted: December 9, 2020

Contents

1 Experiment 1

1.1 Theory

Frequency Filtering has many applications related to image restoration and enhancement. One application in particular is the removal of periodic noise within an image using some type of band-reject filter. This type of filter can be used to zero out a certain range of frequency values within an image. The ideal band-reject filter can be expressed mathematically as

$$H(u,v) = \begin{cases} 0 & D_0 - \frac{W}{2} \leq D \leq D_0 + \frac{W}{2} \\ 1 & otherwise \end{cases}$$

where D is the distance from the center of the band, W is the with of the band, and D_0 is the cutoff frequency.

One issue with using the ideal band-reject filter is the inclusion of a ringing effect due to the overestimate of the frequency of certain image features. Therefore, in order to mitigate such effects a smoothed version of the ideal band-reject filter can be used. One example includes the Butterworth filter, which is expressed as

$$H(u,v) = \frac{1}{1 + \left[\frac{DW}{D^2 - D_0^2}\right]^{2n}}$$

where D, W, and D_0 are as defined before and n is the *order* of the filter. Another example of a smooth band reject filter is the Gaussian filter, defined as

$$H(u,v) = 1 - \exp\left(-\left[\frac{D^2 - D_0^2}{DW}\right]^2\right)$$

These filters can also be used to isolate a specific range of frequency values, which can be used to extract the noise from an image. The filter used in such an operation can be defined using

$$H_{BP}(u,v) = 1 - H(u,v)$$

One example of noise is additive cosine noise of the form $cos(\omega_x x + \omega_y y)$ for some a, b. The noisy image is then expressed as

$$f_{noisy}(x,y) = f(x,y) + cos(\omega_x x + \omega_y y).$$

When taking the Fourier Transform of the noisy image, we can make use of the linearity of the Fourier Transform operation to decouple the spectrums of the original image and the noise,

$$
\begin{aligned}
\mathcal{F}\{f_{noisy}(x,y)\}(u,v) &= \mathcal{F}\{f(x,y)\}(u,v) + \mathcal{F}\{\cos(\omega_x x + \omega_y y)\}(u,v),\\
\mathcal{F}\{cos(\omega_x x + \omega_y y)\}(u,v) &= \mathcal{F}_y\{\mathcal{F}_x\{\cos(\omega_x x + \omega_y y)\}(u,y)\}(u,v)\\
&= \mathcal{F}_y\Big\{\mathcal{F}_x\Big\{\cos\Big(\omega_x\Big(x + \frac{\omega_y}{\omega_x}y\Big)\Big)\Big\}(u,y)\Big\}(u,v)\\
&= \mathcal{F}_y\Big\{\exp\Big(2\pi i\frac{\omega_y}{\omega_x}uy\Big)\mathcal{F}\{\cos(\omega_x x)\}(u)\Big\}(u,v)\\
&= \mathcal{F}_y\Big\{\exp\Big(2\pi i\frac{\omega_y}{\omega_x}uy\Big)\frac{1}{2}\Big[\delta\Big(u-\frac{\omega_x}{2\pi}\Big)+\delta\Big(u+\frac{\omega_x}{2\pi}\Big)\Big]\Big\}(u,v)\\
&= \frac{1}{2}\Big[\delta\Big(u-\frac{\omega_x}{2\pi}\Big)+\delta\Big(u+\frac{\omega_x}{2\pi}\Big)\Big]\mathcal{F}_y\Big\{\exp\Big(2\pi i\frac{\omega_y}{\omega_x}uy\Big)\Big\}(u,v)\\
&= \frac{1}{2}\Big[\delta\Big(u-\frac{\omega_x}{2\pi}\Big)+\delta\Big(u+\frac{\omega_x}{2\pi}\Big)\Big]\delta\Big(v-\frac{\omega_y}{\omega_x}u\Big),
\end{aligned}
$$

so we expect to find impulses at $u = \pm\frac{\omega_x}{2\pi}$ and $v = \frac{\omega_y}{\omega_x}u = \pm\frac{\omega_y}{2\pi}$. To remove the noise, we simply remove these impulses.

1.2 Implementation

For the program implementation, the input image was read using command line arguments, and a copy of the image was made for the various output images. The filtering algorithm was used on two of the input image copies in order to remove or isolate the noise within the original image. Gaussian smoothing was then performed using a similar algorithm from Programming Assignment 2. The resulting images were then saved based on the output file command line argument.

The algorithm for performing the noise removal first involves taking the 2D Fourier Transform of the original image. In order to identify the anomalous frequencies within the image, the spectrum was first shifted by multiplying each pixel by $(-1)^{x+y}$. The complex values generated from taking the FFT of the image were stored using a one dimensional array of values with type `std::complex<float>`. Next, the resulting values were iterated over and the four pixels corresponding to the anomalous frequencies were zeroed out. The decision regarding which frequencies were removed was based on visual analysis of the spectrum of the noisy image. After taking the inverse fourier transform of the modified spectrum, the image pixels were updated by taking the real value and undoing the spectrum shift by multiplying $f(x,y)$ by $(-1)^{x+y}$.

1.3 Results and Discussion

Figure 1 shows the original image, along with its spectrum. The spectrum was generated using the 2D FFT and logarithmic scaling was used in order to enhance visualization. Based on the spectrum, there appears to be four anomalous frequencies forming a rectangle around the center of the spectrum, corresponding to the frequencies generated by the added cosine noise.

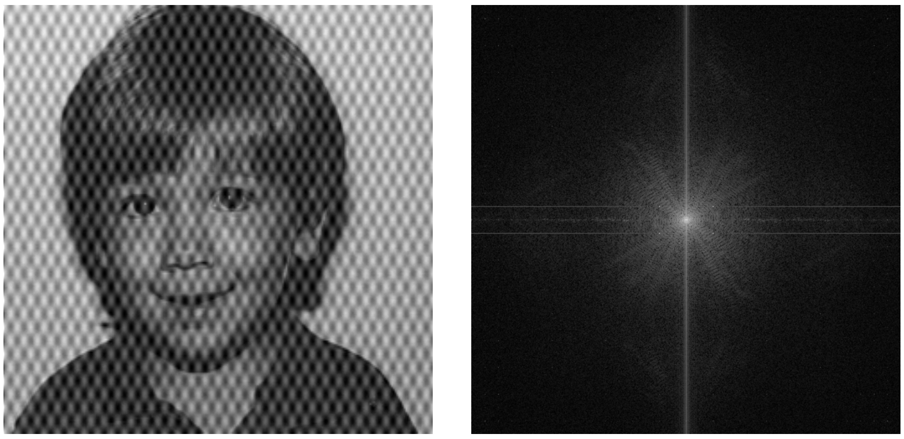


Figure 1: The original noisy image along with its spectrum. The anomalous frequencies can be seen in a grid-like pattern spanning across the spectrum.

he result of zeroing out the anomalous frequency values are demonstrated in fig. 2. Based on the figure, it appears all of the noise has been removed, and the noise could also be isolated from the image rather effectively. For comparison purposes, Gaussian smoothing using both a 7×7 mask and 15×15 mask was performed on the original image. It appears the 15×15 mask did better at reducing the visibility of the noise, however many details within the original image were also lost due to smoothing. The noise pattern was also successfully extracted, and can also be seen in fig. 2.

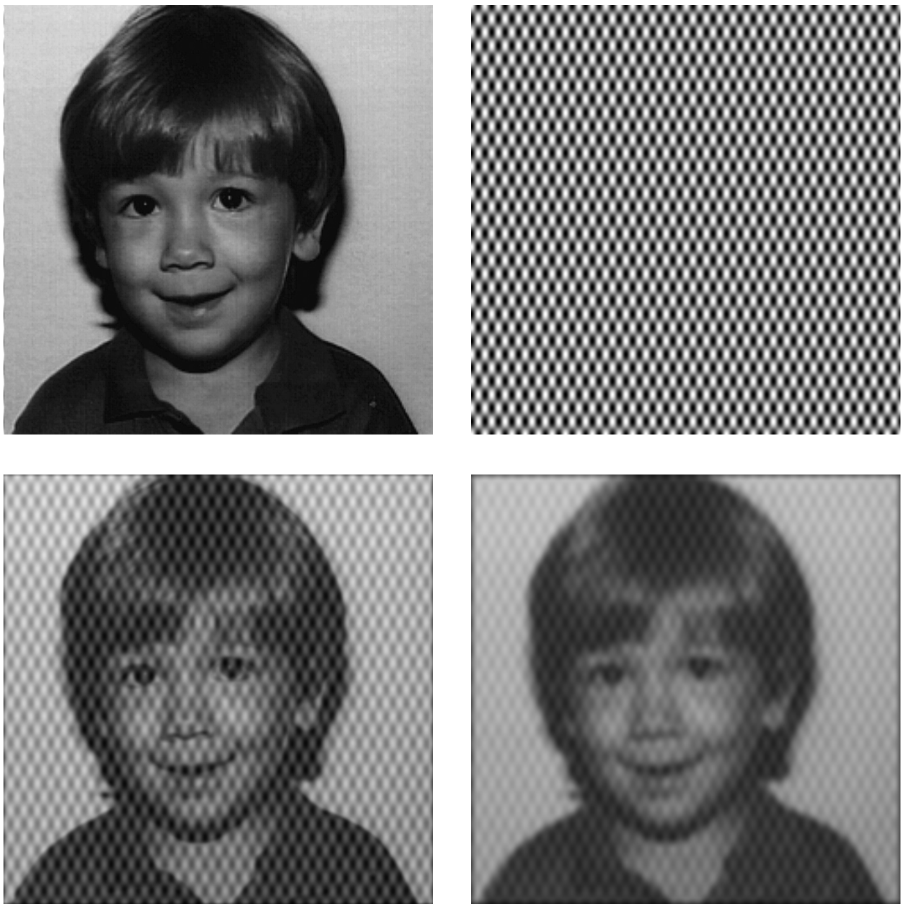


Figure 2: The results of noise removal using frequency filtering (top-left) and noise extraction (top-right). 7×7 (bottom-left) and 15×15 (bottom-right) Gaussian smoothing was also used for comparison.

The effects of noise in images can have extensive implications regarding the utilization and effectiveness of facial verification and facial recognition systems. Failures of such systems can have negative impacts ranging from economic repercussions to privacy and safety concerns. One example involves the use of a facial verification system in commercial products, such as a phone or similar device. If the system used does not account for any noise or anomalous artifacts, it might lead to low performance accuracy. If these issues are widespread, it may lead to economic losses for the company. The failure of facial verification systems also has implications with the privacy of individuals, and ensuring that the system is accurate will help prevent any unauthenticated users from accessing other user's protected information.

The failure of facial recognition systems in the context of law enforcement may also have negative implications regarding public safety and security. With a growing number of law enforcement agencies employing facial recognition systems for identifying criminals, the failure to remove noise and artifacts becomes an even greater concern to safety. In this context, it is important to ensure the accuracy of these systems remains high in order to mitigate the chance of criminals escaping detection, or prevent an innocent person from being wrongly accused. With facial recognition systems being deployed in many public transport systems, it also becomes important to mitigate detection errors in order to help protect the public against potential threats of terrorism. Ensuring proper facial recognition may also prevent economic losses due to the occurrence of a domestic terror attack.

2 Experiment 2

2.1 Theory

From the convolution theorem, we know that convolving two functions in the spatial domain is the same is the same as point-wise multiplication in the frequency domain, i.e.

$$f(x,y) \star g(x,y) = \mathcal{F}^{-1}\{\mathcal{F}\{f(x,y)\}(u,v)\mathcal{F}\{g(x,y)\}(u,v)\}(x,y).$$

We also know that this holds true for sampled functions (multiplied by trains of impulses) and discrete convolution as long as the functions are appropriately padded. However, since our input image dimensions are even and the sobel mask dimensions are odd, there is some ambiguity as to how to how to pad the functions to be the same size, since you can't simply add a beginning and end column/row of zeroes - this would not change parity. According to the book (this does not seem to be shown anywhere) - this important quality to preserve is symmetries - in this case, the odd symmetry property $f(x,y) = -f(M-x, N-y)$, which can be preserved by pre-pending a row and column of zeroes. I personally find the information on this a bit lacking - specifically that according to this definition of odd symmetry, the sobel mask isn't even considered odd, so what is there to preserve?

2.2 Implementation

Spatial filtering is performed as normal (see Programming Assignment 2) by convolving a sobel mask (table 1).

-1	0	1
-2	0	2
-1	0	1

Table 1: Sobel f_x mask.

Frequency filtering is then performed by inserting the sobel mask into a 0-mask the same size of the image, taking care to preserve the proper center. Note that extra padding to avoid wraparound error is not performed, since behavior when convolving at the edges of the image is arbitrarily defined, anyway. Then we find the Fourier Transform of both the image and the mask (not applying a frequency-shift to center either one, since they are in phase with each other regardless), as in previous Programming Assignments. Then, since the sobel mask is applied from the center of the mask, but the center is not stored in $(0, 0)$, a spacial-shift transformation is applied in a similar way to frequency shifts (by multiplying by $(-1)^{u+v}$) due to the symmetry of the Fourier Transform. Finally, the mask transform and the image transform are point-wise multiplied and the inverse Fourier Transform is applied, finishing the filtering process.

2.3 Results and Discussion

Since the sobel mask is real and odd, and we have padded in such a way as to preserve this quality (see section 2.1), the mask's transform is purely imaginary and odd as well. This can be observed in fig. 3.

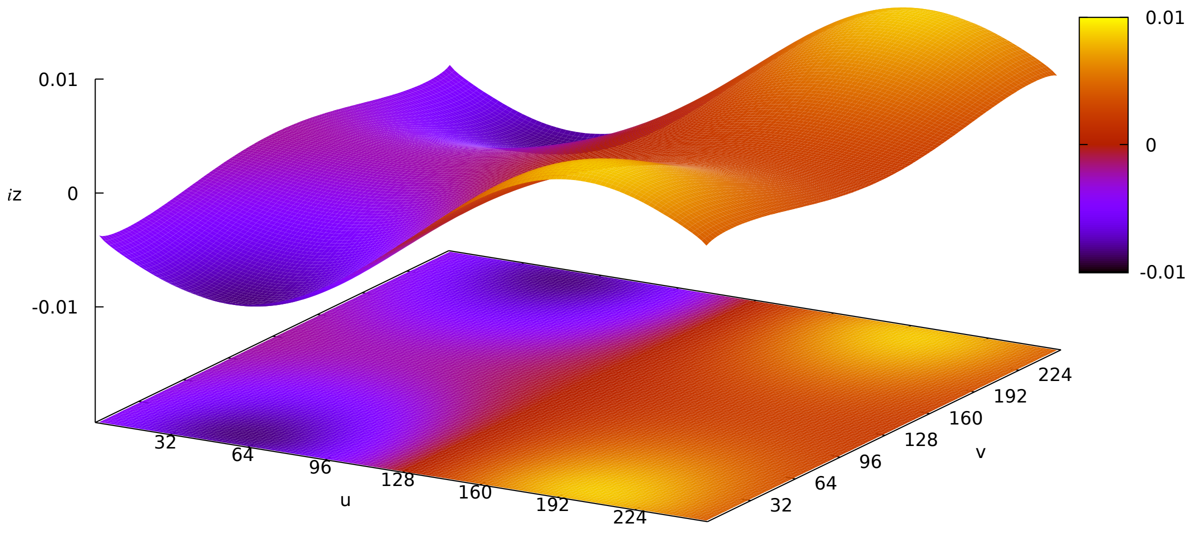


Figure 3: The Fourier Transform of the sobel mask, which is purely imaginary. Note that, since the shift transformation was not applied before taking the transform, it does not look identical to the book.

A comparison of results from filtering in the spatial domain and the frequency domain can be found in fig. 4. Differences can be found at the edges (as discussed in section 2.2), but these are negligeable, and the images are otherwise identical.

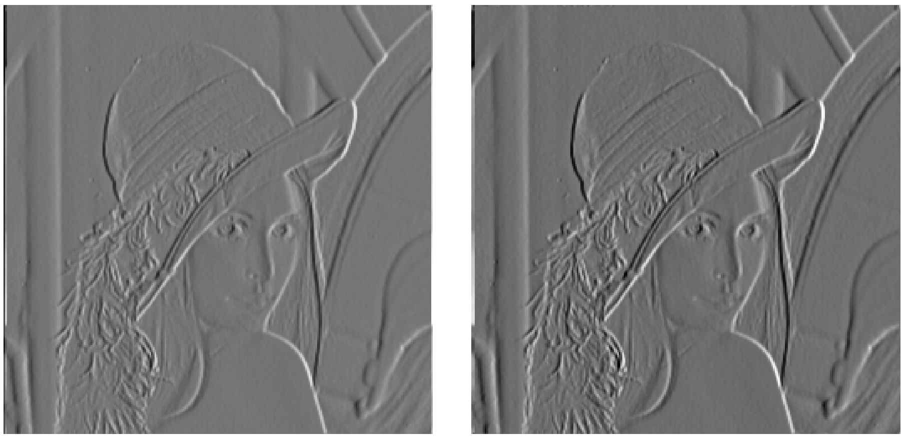


Figure 4: A comparison of spatial filtering (left) and frequency filtering (right) results from filtering the `lenna.pgm` image.

3 Experiment 3

3.1 Theory

One way to model light in a scene is

$$f(x,y) = i(x,y)r(x,y),$$

where $i(x,y)$ is *illumination* (the light emitted by the light source) and $r(x,y)$ is *reflectance* (the percentage of light reflected by an object). The illumination component of a scene changes slowly over the scene and is therefore associated with low frequencies, while the reflectance component changes rapidly (at boundaries of materials and objects) and is therefore associated with high frequencies. In this respect, to highlight objects in the scene, we wish to emphasize the reflectance component and de-emphasize the illumination component. Since the Fourier Transform is a linear operator, we can do this by first taking the natural logarithm of f to separate the two components by addition before taking the Fourier Transform (ass seen in eq. (1)) and applying a High-Pass or similar filter.

$$\begin{aligned} H(u,v)\mathcal{F}\{\ln(f(x,y))\}(u,v) &= H(u,v)\mathcal{F}\{\ln(i(x,y)r(x,y))\}(u,v) \\ &= H(u,v)\mathcal{F}\{\ln(i(x,y)) + \ln(r(x,y))\}(u,v) \\ &= H(u,v)\mathcal{F}\{\ln(i(x,y))\} + H(u,v)\mathcal{F}\{\ln(r(x,y))\}(u,v) \end{aligned} \tag{1}$$

3.2 Implementation

First, a logarithmic transformation is applied to the input image (as discussed in section 3.1). Then, the Fourier Transform is taken as in previous Programming Assignments, with the frequency-shift transformation applied as well (to center it in the frequency domain). Then, the mask $H(u,v)$ (given below) is applied.

$$H(u,v) = (\gamma_H - \gamma_L)\left[1 - \exp\left(-c\left[\frac{u^2+v^2}{D_0^2}\right]\right)\right] + \gamma_L \tag{2}$$

This mask is a high-frequency *emphasis* filter (rather than a high-pass filter), which means it simply emphasizes high frequencies $|(u,v)| > D_0$ by multiplying them by a real scalar $\gamma_H > 1$ and de-emphasizes low frequencies $|(u, v)| < D_0$ by multiplying them by a real scalar $\gamma_L < 1$. Then, c is a "smoothing" constant which controls how sharply the frequencies around D_0 change. This behavior can be seen figs. 5 and 6.

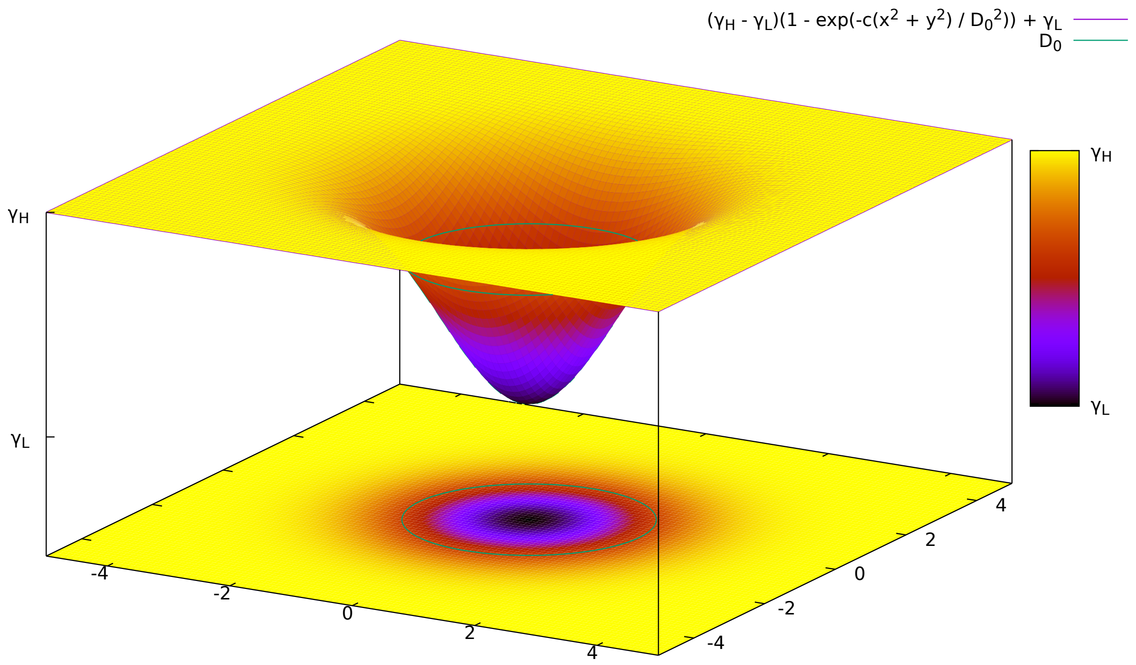


Figure 5: A plot of eq. (2). Note the similarities to a high-pass filter, and that for $\gamma_L = 0$, this filter is a high-pass filter. For this plot $c = 1$.

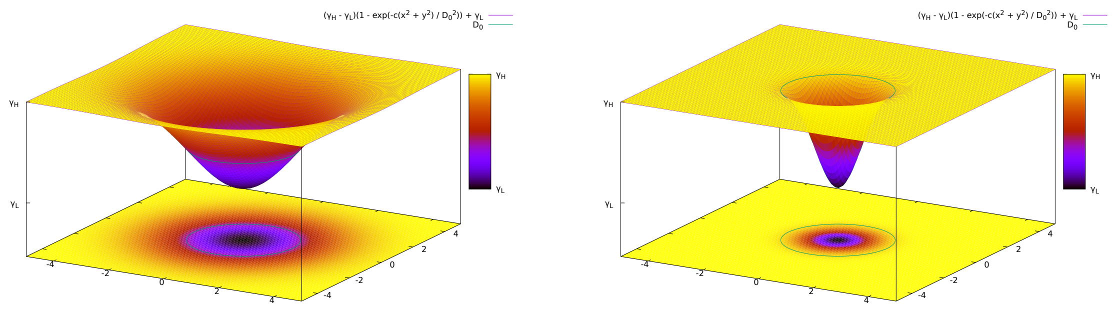
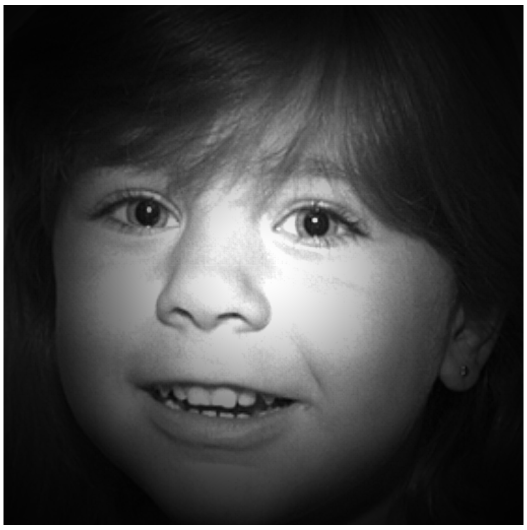


Figure 6: More examples of eq. (2), this time with $c = .5$ (left) and $c = 3.0$ (right) to show the effects of sharpening the filter.

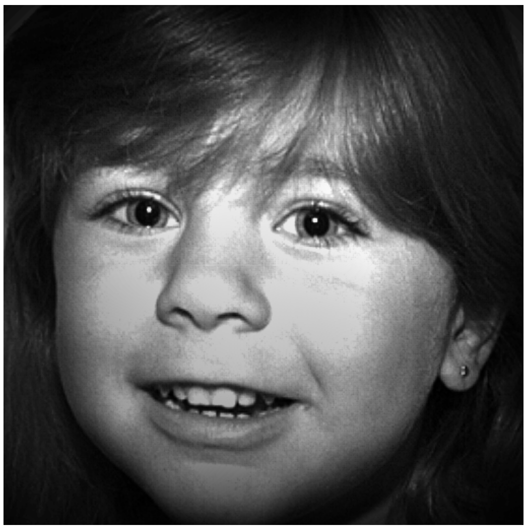
Finally, the inverse Fourier Transform is applied, and an exponential transformation is applied to undo the effects of the natural logarithm applied at the beginning of the process.

3.3 Results and Discussion

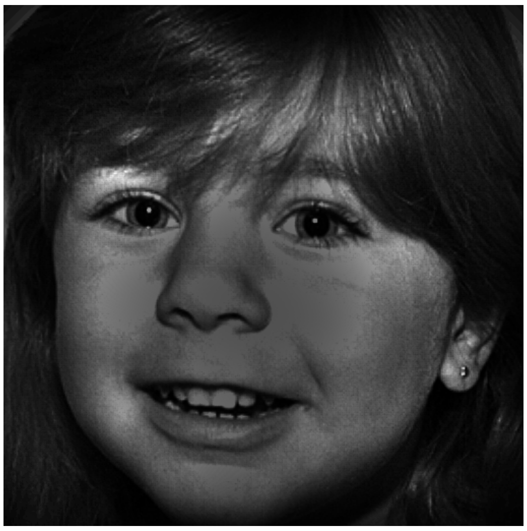
Figure 7 shows the results of the different coefficients. For each image, $D_0 = 1.8$ and $c = 1$ were chosen. The image made with $\gamma_L = 0.5, \gamma_H = 1.5$ does the best at improving contrast.



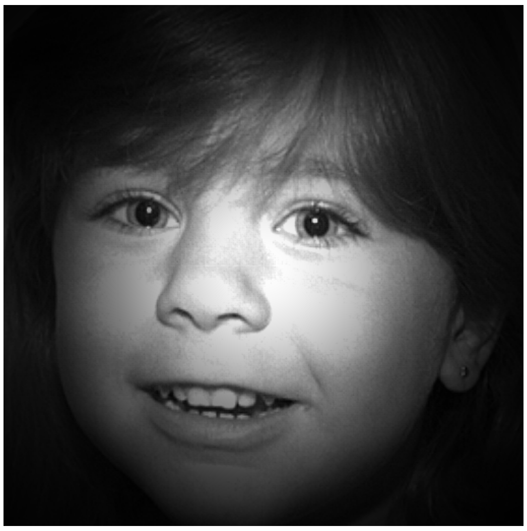
(a) The original image.



(b) Filtered with $\gamma_L = 0.5, \gamma_H = 1.5$.



(c) Filtered with $\gamma_L = 0, \gamma_H = 1$ - a high-pass filter.



(d) Filtered with $\gamma_L = 1, \gamma_H = 1$ - no filter.

Figure 7: A comparison of results of filtering `girl.gif` with various different coefficients.