Solution 1

(a) Ignoring clipping, the observed image intensity can be expressed as

$$I = gI^0 + g\sqrt{I^0}\epsilon_1 + \sqrt{g^2\sigma_{2a}^2 + \sigma_{2b}^2}\epsilon_2 \tag{1}$$

Since both ϵ_1 and ϵ_2 are $N(0,1)$ Gaussian random variables, we can express the two random terms as the following.

$$g\sqrt{I^0}\epsilon_1 \sim N(0, g^2 I^0) \tag{2}$$

$$\sqrt{g^2\sigma_{2a}^2 + \sigma_{2b}^2}\epsilon_2 \sim N(0, g^2\sigma_{2a}^2 + \sigma_{2b}^2) \tag{3}$$

The variance of the sum of two random normal variables is simply the sum of the variances. Thus,

$$\text{Var}(\epsilon) = g^2 I^0 + g^2\sigma_{2a}^2 + \sigma_{2b}^2 \tag{4}$$

(b) Since the amplification factor g is now k times bigger, the variance will grow by a factor of k^2. Thus,

$$\text{Var}(\epsilon) = k^2 * (g^2 I^0 + g^2\sigma_{2a}^2 + \sigma_{2b}^2) \tag{5}$$

(c) Each I_n for $n \in 1...k$ is a Gaussian random variable with variance $k^2 * (g^2 I^0 + g^2\sigma_{2a}^2 + \sigma_{2b}^2)$.
The intensity I can be defined as below.

$$I = \frac{\sum_{i=0}^{k} I_i}{k} \tag{6}$$

Thus, the variance of the sum of the k intensities is as follows:

$$k * \text{Var}(\epsilon) = k^3 * (g^2 I^0 + g^2\sigma_{2a}^2 + \sigma_{2b}^2) \tag{7}$$

Next applying the division by k results in the following result

$$\text{Var}(I) = \frac{k * \text{Var}(\epsilon)}{k^2} = k * (g^2 I^0 + g^2\sigma_{2a}^2 + \sigma_{2b}^2) \tag{8}$$

(d) Taking one shot with exposure time T would be preferable since taking k shots with T/k exposure time would result in k times more noise, as shown by equation 8.

Solution 2

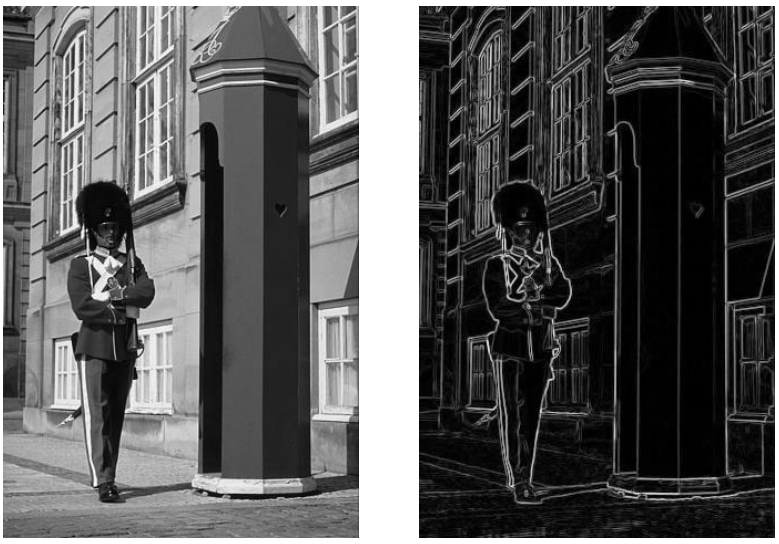
Figure 1 shows the results of the histogram equalization algorithm implemented in `code/prob2.py`.

(a) Input image (b) Image after histogram equalization

Figure 1: Input image before and after histogram equalization

Solution 3

(a) Using just the Sobel kernels to calculate the gradients, the following image was generated.



(a) Input image (b) Output image

Figure 2: Input image and Gradient magnitude of image calculated using Sobel kernels

(b) After implementing non-maxima suppression, a thresholds of 0.5 seemed to give good results in detecting edges. The results before and after non-maxima suppression are shown below.

(a) Output image with a threshold of 1 without Non-maxima suppression (b) Output image with a threshold of 1 with Non-maxima suppression

Figure 3: Output images with and without Non-maxima suppression

Solution 4

(a) Input image with less noise

(b) Input image with more noise

Figure 4: Input image and Gradient magnitude of image calculated using Sobel kernels

Figure 5: Output image from bilateral filtered with $\sigma_s = 2$, $sigma_i = 0.5$

Figure 6: Output image from bilateral filtered with $\sigma_s = 4$, $sigma_i = 0.25$

Figure 7: Output image from bilateral filtered with $\sigma_s = 16$, $sigma_i = 0.125$

Figure 8: Output image from bilateral filtered with $\sigma_s = 2$, $sigma_i = 0.125$, repeated 8 times

Figure 9: Output image from bilateral filtered with $\sigma_s = 2$, $sigma_i = 0.125$, repeated 8 times. This was generated using the noisier input image

Solution 5

(a) The Fourier transform of an image of width W and height H only needs to store WH scalars due to $F[-u,-v] = F[W-u, H-v] = \bar{F}[u,v]$.

Because of the second property, the reflected value of a pixel is just its conjugate. This means that the pixel at $(W-u, H-v)$ is just the conjugate of (u,v).
When applying this concept, the first row and column must be treated separately, since $F[u,0] = \bar{F}[-u,0]$, which is in the same column, and similarly with rows. However, this still means that half the values need to be store, since the other half is just the complex conjugate.
The final property to note is that $F[0,0]$ is the DC component and is always real if X is real. Thus, this will add 1 additional distinct value. However, these properties depends on the parity of W and H. In order to prove this, we will treat them separately.

- If both W and H are odd, then the first (index 0) column and row will have an even number of values when excluding the DC component ($W-1$ and $H-1$ are even). Thus, these will contribute $\frac{W-1}{2}$ and $\frac{H-1}{2}$ unique complex values.
  Since both W and H are odd, then the rest of the values can be divided in half (along either rows or columns) and the symmetry property can be used to recover the other half. Thus, this part contributes $\frac{(W-1)(H-1)}{2}$ unique complex values.
  Overall, there are $\frac{W-1}{2} + \frac{H-1}{2} + \frac{(W-1)(H-1)}{2} = \frac{WH-1}{2}$ unique complex values. Since each complex value is 2 unique scalars, and adding the DC component, which is 1 unique scalar, we get $2 * \frac{WH-1}{2} + 1 = WH$ unique scalars

- If both W and H are even, then first (index 0) column and row will have an odd number of values when excluding the DC component ($W-1$ and $H-1$ are odd). An additional property that must be leveraged which is that $F[W/2,0]$, $F[0,H/2]$, and $F[W/2,H/2]$ are all real-valued. This is due to $F[-W/2,0] = F[W-W/2] = F[W/2,0] = \bar{F}[W/2,0]$. The only way for the conjugate to equal the non-conjugate value is for the imaginary part to be zero. Similar logic can be applied to $F[0,H/2]$ and $F[W/2,H/2]$.
  Both the first column and row (excluding DC) contribute $\frac{W-2}{2}$ and $\frac{H-2}{2}$ unique complex numbers, as well as 1 unique real valued number. Furthermore, the rest of the Fourier transform is $(W-1)(H-1)$ elements, which is odd. However, all values will be symmetric about $F[W/2,H/2]$, which is real valued. Thus, the remaining values will contribute $\frac{(W-1)(H-1)-1}{2}$ unique complex numbers, and 1 unique real valued number.
  Combining these we get $\frac{W-2}{2} + \frac{H-2}{2} + \frac{(W-1)(H-1)-1}{2} = \frac{WH-4}{2}$ unique complex numbers, and 4 unique real valued numbers (including the DC component). Since each complex number is two scalars, there are $2 * \frac{WH-4}{2} + 4 = WH - 4 + 4 = WH$ unique scalars.

- If one of W or H is even and the other is odd, then one of $W-1$ and $H-1$ is even and the other is odd. For the rest of this proof, we will use H as even and W as odd, but the variables could simply be swapped to prove the other case.
  Since $W-1$ is even, the first row has $\frac{W-1}{2}$ unique complex numbers. Since $H-1$ is odd, the first column has $\frac{W-2}{2}$ unique complex numbers, as well as 1 real valued number, as shown in the even-even case. The rest of the Fourier transform contains $(W-1)(H-1)$ elements, which is even, and can thus be split into two groups and can leverage symmetry. As such, this section contributes $\frac{(W-1)(H-1)}{2}$ unique complex numbers.
  Combining the above, we get $\frac{W-1}{2} + \frac{H-2}{2} + \frac{(W-1)(H-1)}{2} = \frac{WH-2}{2}$ unique complex numbers and 2 unique real valued numbers. Since each complex number is 2 scalars, we will get $2 * \frac{WH-2}{2} + 2 = WH$ unique scalars.

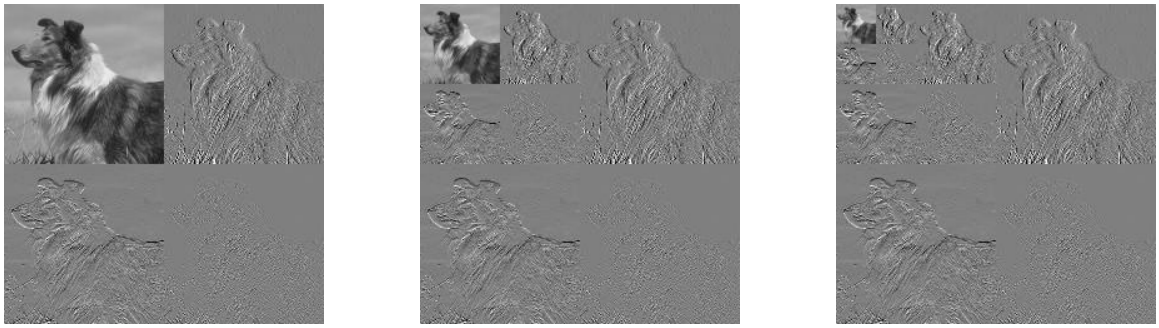
Using this fact, only WH scalars need to be stored in any case, since there are only WH unique scalars, and the rest either are complex conjugates, or zero (as in the even-even case).

(b) The following image was generated using standard convolution, as well as convolution in Fourier domain.

Figure 10: Images from left to right: Input, Gaussian convolution using convolution in standard domain, Gaussian convolution implemented using Fourier domain

Solution 6

(a) Figure 11 shows the wavelet transform at various levels for a gray scale image of a dog.



(a) Haar wavelet at level 1 (b) Haar wavelet at level 2 (c) Haar wavelet at level 3

Figure 11: Haar Wavelet pyramid

(b) Figure 12 uses shows using the wavelet transform to recreate the original image.

Figure 12: Reconstructed image from inverse Haar wavelet

Figure 13 shows the same reconstruction, but with some layers zeroed out.

(a) Haar wavelet reconstruction with one layer removed (b) Haar wavelet reconstruction with two layers removed (c) Haar wavelet reconstruction with three layers removed

Figure 13: Haar Wavelet reconstruction from the pyramid

Information

This problem set took approximately 12 hours of effort.

I discussed this problem set with:

- Myself

I also got hints from the following sources:

- Read numpy documentation about `np.unique` https://numpy.org/doc/stable/reference/generated/numpy.unique.html
- For rolling matrices reference https://stackoverflow.com/questions/19878280/efficient-way-to-shift-2d-matrices-in-both-directions
- Scipy Convolve2d documentation https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html
- Numpy documentation for `np.pad` https://numpy.org/doc/stable/reference/generated/numpy.pad.html