Solution 1

(a) Given the cost function

$$(x-y)^2 + \lambda|x|$$

It's derivative in terms of x is as follows

$$2x - 2y + \lambda * sgn(x)$$

where $sgn(x)$ is the sign function and returns +1 if $x > 0$ and -1 if $x < 0$. This gives the following values for x:

$$x = y - \lambda$$

$$x = y + \lambda$$

As such, we have two values that can possibly minimize the cost function, where we will simply take the smallest of the two possible values. Additionally, since the gradient of the absolute value function is discontinuous at 0, we must also compare with the value at $x = 0$.

(b) The output to a Python implementation of denoising using L1 regularization is shown below in 1.

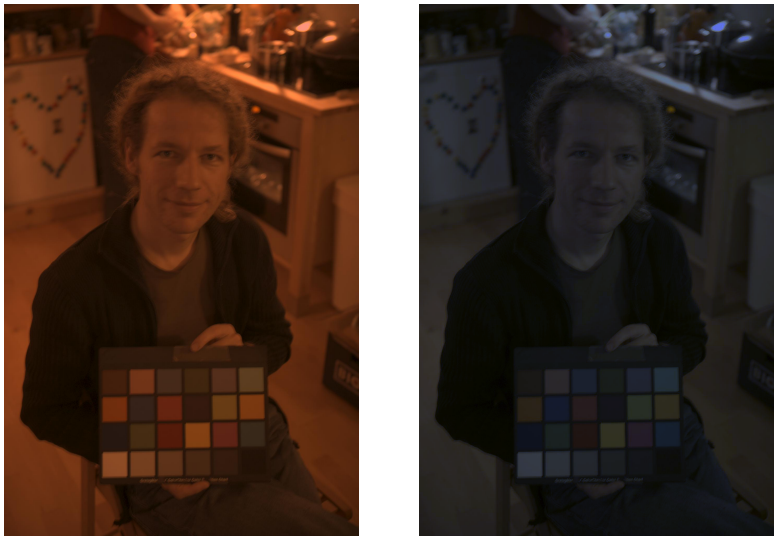
Figure 1: Denoised image using L1 regularization

Solution 2

(a) Figures 2, 3, and 4 show white balanced images using the Gray world assumption. These images tend to be more gray at output.

(a) Input image (b) Output image

Figure 2: White balanced image using Gray world assumption



(a) Input image (b) Output image

Figure 3: White balanced image using Gray world assumption

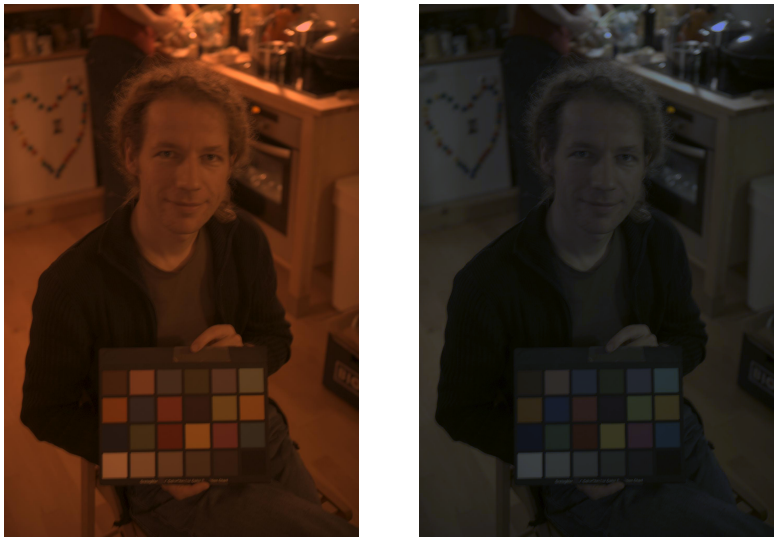
(a) Input image (b) Output image

Figure 4: White balanced image using Gray world assumption

(b) Figures 5, 6, and 7 show white balanced images computed using the average of the top 10% brightest intensities on each channel. This method of white balancing can lead to amplifying one color, such as in 6.

(a) Input image (b) Output image

Figure 5: White balanced image using 10% brightest intensities



(a) Input image (b) Output image

Figure 6: White balanced image using 10% brightest intensities

(a) Input image (b) Output image

Figure 7: White balanced image using 10% brightest intensities

Solution 3

(a) The estimated normals using least squares regression is shown in 8.

Figure 8: Estimated normals

(b) The albedo of the image was estimated using the same least squares regression method is shown below in 9

Figure 9: Estimated albedos

Solution 4

The depth map of generated from the estimated surface normals using the Frankot-Chellappa method is shown below in 10

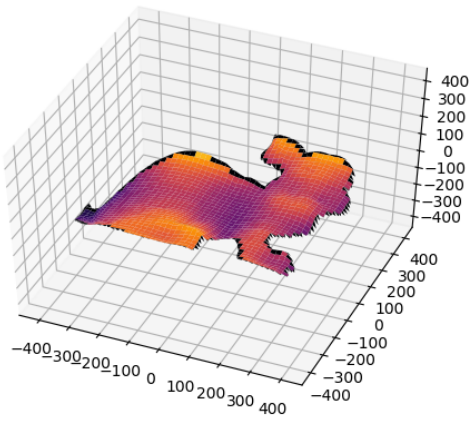


Figure 10: Depth map computed using the Frankot-Chellappa method.

Solution 5

The depth map of generated from the estimated surface normals using the conjugate gradient method is shown below in 11. For this image, 100 iterations were used.

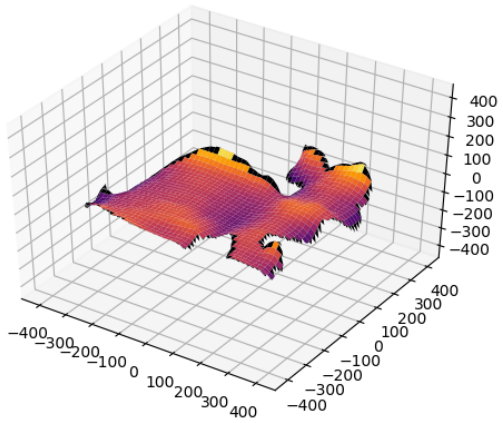


Figure 11: Depth map computed using 100 iterations conjugate gradients.

Information

This problem set took approximately 20 hours of effort.

I discussed this problem set with only myself

I also got hints from the following sources:

- Wikipedia article on matrix calculus at https://en.wikipedia.org/wiki/Matrix_calculus
- Read numpy tutorial from https://docs.scipy.org/doc/numpy-1.13.0/user/basics.broadcasting.html
- Read numpy documentation for `np.linalg.solve` https://numpy.org/doc/stable/reference/generated/numpy.linalg.solve.html?highl
- Read numpy documentation for `np.conjugate` https://numpy.org/doc/stable/reference/generated/numpy.conjugate.html?highlight=c
- Documentation for `scipy.convolve2d` https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html