Solution 1

(a) To find the minimum over x of

$$f(x) = (x-y)^2 + \lambda|x| \tag{1}$$

we start by proving that $f(x)$ is a convex function over $\mathcal{R}$ for $\lambda \geq 0$. To do this, we will use a theorem, $f(x)$ is convex iff

$$f(a) \geq f(b) + \partial f(b)(a-b) \quad \forall x, y \in \mathcal{R} \tag{2}$$

where ∂f is defined to be the subgradient of f. To start, we compute the subgradient of the non-differentiable function:

$$\partial f(x) = 2(x-y) + \begin{cases} \lambda & \text{for} \quad x > 0 \\ \lambda[-1,1] & \text{for} \quad x = 0 \\ -\lambda & \text{for} \quad x > 0 \end{cases} \tag{3}$$

We then plug this in to the theorem above, and show that it holds. We get

$$(a-y)^2 + \lambda|a| \geq (b-y)^2 + \lambda|b| + 2(b-y)(a-b) + (a-b) \begin{cases} \lambda & \text{for} \quad b > 0 \\ \lambda[-1,1] & \text{for} \quad b = 0 \\ -\lambda & \text{for} \quad b > 0 \end{cases} \tag{4}$$

Rearranging, and canceling terms we simplify it to

$$(a-b)^2 + \lambda(|a| - |b|) \geq \begin{cases} \lambda(a-b) & \text{for} \quad b > 0 \\ \lambda a[-1,1] & \text{for} \quad b = 0 \\ -\lambda(a-b) & \text{for} \quad b > 0 \end{cases} \tag{5}$$

Considering each case individually, we get three inequalities that must hold:

$$(a-b)^2 + \lambda(|a| - |b|) \geq \lambda(a-b) \quad \text{for} \quad b > 0 \tag{6}$$

$$(a-b)^2 + \lambda(|a| - |b|) \geq \lambda a[-1,1] \quad \text{for} \quad b = 0 \tag{7}$$

$$(a-b)^2 + \lambda(|a| - |b|) \geq -\lambda(a-b) \quad \text{for} \quad b < 0 \tag{8}$$

For the first inequality, we rewrite it as

$$(a-b)^2 + \lambda(|a| - a) - \lambda(|b| - b) \geq 0 \quad \text{for} \quad b > 0 \tag{9}$$

and note that every term is greater than or equal to zero. The absolute value of a scalar subtracted by itself can be

$$|a| - a = \begin{cases} 0 & \text{for} \quad a \geq 0 \\ -2a & \text{for} \quad a < 0 \end{cases} \tag{10}$$

Since we assume $b > 0$, $|b| - b = 0$, and $|a| - a \geq 0$ for any a, the inequality holds. For the second case, we simply set $b = 0$ for all terms to get

$$(a)^2 + \lambda(|a| - a[-1,1]) \geq 0 \tag{11}$$

and note that $a[-1,1] \leq |a|$, since the maximum of the range is $|a|$. Thus the inequality holds. For the third inequality we rewrite it similar to equation 9,

$$(a-b)^2 + \lambda(|a| + a) - \lambda(b + |b|) \geq 0 \quad \text{for} \quad b < 0 \tag{12}$$

which we rewrite similarly to equation 9, and note that,

$$|a| + a = \begin{cases} 2a & \text{for} \quad a \geq 0 \\ 0 & \text{for} \quad a < 0 \end{cases} \tag{13}$$

Since we assume $b < 0$, $|b| + b = 0$, and every other term in the inequality is positive. Since the inequality for the theorem described in equation 2 holds, we conclude that f is convex on the interval of $\mathcal{R}$. From this, we know that f has a global minimum at x^* satisfying $0 \in \partial f(x^*)$. Using the subderivative defined in equation 3, we define three cases, based on λ and y:

$$x^* = \begin{cases} y - \frac{\lambda}{2} & \text{for} \quad y > \frac{\lambda}{2} \\ 0 & \text{for} \quad -\left|\frac{\lambda}{2}\right| \leq y \leq \left|\frac{\lambda}{2}\right| \\ y + \frac{\lambda}{2} & \text{for} \quad y < -\frac{\lambda}{2} \end{cases} \tag{14}$$

with x^* as the unique minimizer.

(b) In this part we make a denoise function that uses the cost function described in part (a), equation 1. To do this, we look at every pixel, and do a boolean test of the conditions in equation 14, and then apply the update conditions to the pixels for each case. The resulting denoised image is shown in Figure 1:

Figure 1: Wave Transform Noise Correction

Solution 2

(a) To balance color using a Gray World assumption, we calculate scalars to multiply each color channel by, penalizing colors that have larger components, by dividing by the mean of the intensity of that color channel based on,

$$Y_i[n] = X_i[n] * \alpha_i \tag{15}$$

where $X_i[n]$ is the intensity of a pixel in the ith color channel, and α_i is the scalar multiplier. We calculate α_i for each channel with

$$\alpha_i = \frac{c}{\mu_i} \tag{16}$$

where μ_i is the mean intensity of the ith color channel. We want to normalize the multipliers such that

$$\alpha_r + \alpha_g + \alpha_b = 3 \tag{17}$$

We do this by solving for the normalizing factor c, getting

$$c = \frac{3}{\frac{1}{\mu_r} + \frac{1}{\mu_g} + \frac{1}{\mu_b}} \tag{18}$$

We can then update each color channel individually using equation 15, or multiply by a matrix

$$Y[n] = \begin{bmatrix} \alpha_r & 0 & 0 \\ 0 & \alpha_g & 0 \\ 0 & 0 & \alpha_b \end{bmatrix} X[n] \tag{19}$$

The result of this color correction is shown in Figure 2.

(b) In this part, we use a different prior to color correct, the White Patch Retinex. This assumes that under canonical illumination, the brightest parts of the image are white. Under this assumption, we only need to look at the brightest pixels in each color channel for our color correction. The algorithm is exactly the same as described in part (a), except for calculating μ_i, we only use the k most intense pixels for each color channel. In this case, we used $k = W_x H_x/10$, to get the mean of the brightest 10% of pixels for each channel. The results of this algorithm are shown in Figure 3.

The first thing to note about the these two priors is that neither preserve the mean intensity of the image. For the three images for the homework, the color correction darkened them all at least a little bit. The Gray World prior struggled with the two images that had overwhelming parts of one color (light brown in the first image and green in the third), washing out the dominant color, while the White Patch dealt with this better. However, the White Patch struggled with correcting the second image, which did not a strong "white" set of reference pixels since it was darker. The reason the White Patch does better with images with a large part of a single color is that it allows the mean intensity across color channels to be different, while Gray World sets the color channels to the same mean intensity.

Figure 2: Gray World Color Correction

Figure 3: White Patch Retinex Color Correction

Solution 3

(a) In this problem we calculate the surface normal of an image using photometric stereo of a Lambertian surface. Since the object is Lambertian, we can use the intensity of the light from a known point source to calculate the angle of the surface relative to the light. With three or more linearly independent light sources, we can use this to calculate the surface normal for all three dimensions. We do this for every pixel in the image independently to get the surface normal of the whole object. We can write the grayscale intensity observed from a single light source, I_i as

$$I_i = \rho \langle \ell_i, \hat{n} \rangle \tag{20}$$

with ℓ_i is a vector describing the vector of a point light source. We can absorb the scalar ρ into $\hat{n}$ since we can renormalize after calculating. Expanding to get an expression for a number of images, we get

$$I = Ln \tag{21}$$

where I is a $k \times 1$ vector of grayscale intensities, L is a $k \times 3$ vector describing each point light source ℓ_i^T, and $n = \rho\hat{n}$, the surface normal scaled by the albedo. Since we assume the light sources are white, we don't have to worry about ρ having different components for different colors. To solve for the normal for the pixel, we use least squares,

$$n = \arg\min_n ||Ln - I||^2 \tag{22}$$

which we can solve using a pseudo-inverse of L in equation 21,

$$n = (L^T L)^{-1} L^T I \tag{23}$$

or a Cholesky solver of the gradient set equal to zero,

$$L^T Ln = L^T I \tag{24}$$

Since we can compute $(L^T L)^{-1} L^T$ a single time, with relative efficiency and $(L^T L)^{-1}$ is guaranteed to exist if we have at least three linearly independent light sources, I used the pseudo-inverse method. Lastly we compute $\hat{n}$ by dividing by the norm of the calculated n. Since we can construct I in matrix form as a $(k \times W_X H_X)$ matrix, we can get the normal for every pixel in a single matrix multiplication of the pseudo-inverse of L times the intensity matrix I. The resulting matrix $\hat{n}$ is a $(3 \times W_X H_X)$ matrix that we transpose and reshape to get the surface norm at every unmasked pixel in the image.The result is shown in Figure 4.

Figure 4: Surface Norm Mapping from Photometric Stereo

(b) In this problem we use the calculated surface normal and information from the photometric stereo images to calculate the color albedo of the object. This is again done on each pixel individually by minimizing the square norm cost function

$$\rho_i = \arg\min_{\rho_i} \sum_k (I_{k,i} - \rho_i \ell_k^T \hat{n})^2 \tag{25}$$

where i represents the ith color channel and k represents the kth image. Taking the gradient and setting it equal to 0, we can rewrite the expression in terms of a matrix sum,

$$I_i^T L\hat{n} = \rho_i (L\hat{n})^T (L\hat{n}) \tag{26}$$

and since $(L\hat{n})^T(L\hat{n}) = ||L\hat{n}||^2$ is a scalar, we just divide it across to get ρ_i,

$$\rho_i = \frac{I_i^T L\hat{n}}{||L\hat{n}||^2} \tag{27}$$

We can do this for each of the color channels of the image, but since the only difference between doing it for a single color channel and for all the color channels is I, we can just use the intensity vector for all three color channels to get ρ for all channels at once:

$$\rho = \frac{I^T L\hat{n}}{||L\hat{n}||^2} \tag{28}$$

We then calculate ρ for each unmasked pixel to get the color albedo vector of the entire object. The result is shown in Figure 5.

Figure 5: Color Albedo from Photometric Stereo

Solution 4

In this problem we compute a depth map, Z, of the object in Problem 3, using a deconvolution of the gradient with the directional derivatives:

$$g_x = Z * \underbrace{[1/2, 0, -1/2]}_{f_x} \tag{29}$$

$$g_y = Z * \underbrace{[-1/2, 0, 1/2]^T}_{f_y} \tag{30}$$

since we have two equations for Z, we use a least squares formulation with a regularization term, $\lambda R(Z) = \lambda \sum_n (Z[n] * f_r)^2$ to solve for Z:

$$Z = \arg\min_Z ||g_x - Z * f_x||^2 + ||g_y - Z * f_y||^2 + \lambda \sum_n (Z[n] * f_r)^2 \tag{31}$$

where f_r is a 3×3 kernel with $f_r[i, j] = -1/9$ for $[i, j] \neq [1, 1]$, and the center being $8/9$. We start by converting to the Fourier domain so we can deal with convolutions as multiplication, and taking the gradient equal to 0 there to get

$$0 = -2(\bar{F}_x G_x + \bar{F}_y G_y) + 2(|F_x|^2 + |F_y|^2 + \lambda |F_r|^2)\mathcal{F}(Z) \tag{32}$$

where F_i is the Fourier Transform of the f_i kernel, G_i is the transform of g_i image, and $\mathcal{F}(Z)$ is the transform of Z and bar denotes complex conjugate. This is a linear equation that we can write a closed form solution for,

$$\mathcal{F}(Z) = \frac{\bar{F}_x G_x + \bar{F}_y G_y}{|F_x|^2 + |F_y|^2 + \lambda |F_r|^2} \tag{33}$$

which we can take the inverse transform of to get our map Z. The map generated by this algorithm is shown in Figure 6

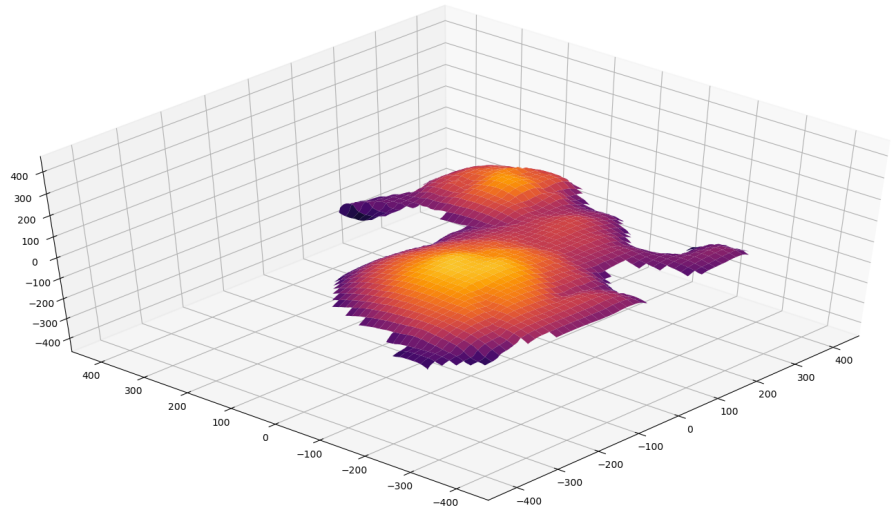

Figure 6: Depth Map from Frankot-Chellappa Method

This algorithm is very quick, because the FFT operation makes taking a Fourier transform very efficient, and takes the place of large convolutions. It becomes infeasible if it is difficult or impossible to form the elements of the solution in Fourier space, for example if you wanted to weight individual pixels.

Solution 5

An alternative to the algorithm in problem 4 is the Conjugate Gradient method, this has the advantage of not having to explicitly form the matrices in Fourier space. This allows us to use pixel-independent weighting, which would create a $W_X H_X \times W_X H_X$ diagonal matrix in Fourier Space. The weighting we used is

$$w[n] = (\hat{n}_z[n])^2 \tag{34}$$

and we want to use the least-squares formulation,

$$Z = \arg\min_Z \sum_n w[n](g_x[n] - (f_x * Z)[n])^2 + \sum_n w[n](g_y[n] - (f_y * Z)[n])^2 + \lambda R(Z) \tag{35}$$

which we can write as a quadratic equation in Z,

$$Z = \arg\min_Z \ Z^T Q Z - 2Z^T b + c \tag{36}$$

which we can use conjugate gradient method to solve without explicitly forming the $W_X H_X \times W_X H_X$ matrix Q. The algorithm describes an initialization

$$r_0 = p_0 = b - Q * Z_0 \tag{37}$$

and iterative update,

$$\begin{aligned} \alpha_k &= \frac{||r_k||^2}{p_k^T Q p_k} \\ Z_{k+1} &= Z_k + \alpha_k p_k \\ r_{k+1} &= r_k - \alpha_k Q p_k \\ \beta_k &= \frac{||r_{k+1}||^2}{||r_k||^2} \\ p_{k+1} &= r_{k+1} + \beta_k p_k \end{aligned} \tag{38}$$

for each iteration of the algorithm $k = \{0,\ 1,\ \ldots\ ,\ K\}$, returning Z_K, where $Z_i,\ r_i,\ p_i$ are image shaped objects and $\alpha_i,\ \beta_i$ are scalars.

From equation 35 we can see that there is no scalar component in the quadratic, and we can use a fourier transform write expressions for b and Q,

$$\mathcal{F}(Q) = F_x^T W F_x + F_y^T W F_y + \lambda F_r^T F_r \tag{39}$$

$$\mathcal{F}(b) = F_x^T W G_x + F_y^T W G_y \tag{40}$$

b we can inverse transform, turning multiplication in transform space into convolutions, since b is an image shaped object,

$$b = (g_x \times w) * \bar{f}_x + (g_y \times w) * \bar{f}_y \tag{41}$$

where $\times$ denotes element-wise multiplication, and $\bar{f}_i$ denotes the flipped kernel of f_i, which comes from the transpose of a kernel F_i^T in Fourier space. Q, as a square matrix denotes a convolution operation, so we can formulate Qp the matrix multiplication of Q with a image shaped vector, p

$$Qp = ((p * f_x) \times w) * \bar{f}_x + ((p * f_y) \times w) * \bar{f}_y + \lambda((p * f_r) * f_r) \tag{42}$$

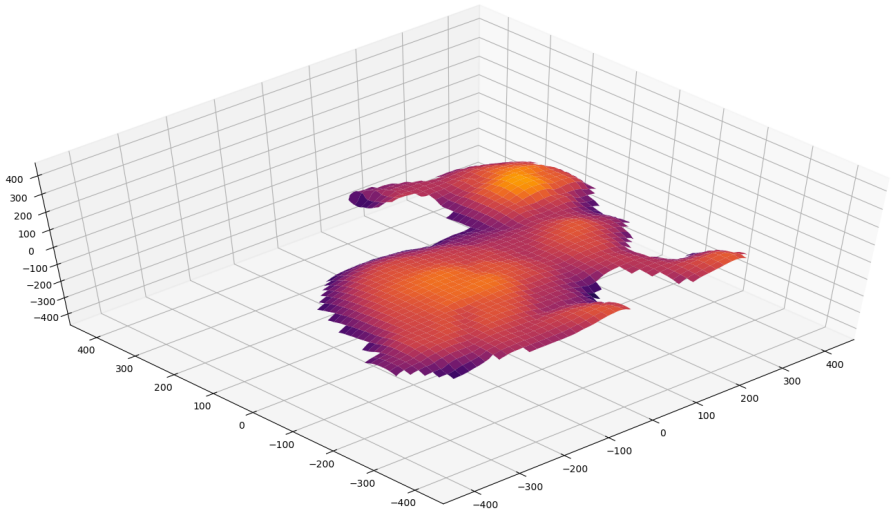

Figure 7: Depth Map from Conjugate Gradient Method

The result of this algorithm is shown in Figure 7.

This algorithm is much more expensive than the Frankot-Chellappa Method, since there are K iterations where you have to compute multiple convolutions across the whole image, instead of only a handful of image sized FFTs and elementwise multiplications, however it has the potential to be accurate since we are able to apply a weight to individual pixels. This can be seen in the differences between Figures 6 and 7, where there are finer details in Figure 7.

Information

This problem set took approximately 20 hours of effort.

I discussed this problem set with only the instructor.

I also got hints from no outside resources.