Computational Photography - Programming exercise #1

Due: April 30, 2022. In pairs

Intro

In this assignment we will implement deconvolution in the primal and frequency domains.

In class we discussed the problem of deblurring an $m \times n$ image y. We assume that the blurred image y is generated as $y = x * k + n$ where k is a blur kernel and $n \sim N(0, \eta^2)$ is a zero mean Gaussian noise. We introduce a prior on natural images

$$p(x) \propto e^{\frac{1}{2\sigma^2}\sum_{i,j}|g^x_{ij}(x)|^2+|g^y_{ij}(x)|^2}$$

where σ is the variance (a parameter of the prior) and $g^x_{ij}(x)$ and $g^y_{ij}(x)$ are the responses at coordinate (i, j) of the image to the filters $(-1, 1)$ and $(-1, 1)^T$ respectively. We then formulated the problem of deblurring using a prior on the distribution of the edges in the image as finding a minimizer of

$$L(x) = \frac{1}{2\eta^2}\|y - Ax\|^2 + \frac{1}{2\sigma^2}\sum_{1\le i\le m, 1\le j\le n}|g^x_{ij}(x)|^2 + |g^y_{ij}(x)|^2 \tag{1}$$

Where A is the $mn \times mn$ blur matrix corresponding to k, y is the column-stacked blurred image and η^2 is the variance of the noise in the blurred image.

In the frequency domain one minimizes

$$L(\mathcal{F}_x(\omega)) = \frac{1}{2\eta^2}\|\mathcal{F}_y(\omega) - \mathcal{F}_k(\omega)\cdot\mathcal{F}_x(\omega)\|^2 + \frac{1}{2\sigma^2}\left(\|\mathcal{F}_{g^x}(\omega)\|^2 + \|\mathcal{F}_{g^y}(\omega)\|^2\right) \tag{2}$$

1 Implementation

You should submit the following function:

1. Primal deconvolution **function est_x=deconvPrimal(y,k,eta,sigma,is_cyclic)** where y is an $m \times n$ image and η and σ are scalars as defined above. is_cyclic is a binary variable denoting if convolution with k, g^x, g^y are cyclic or not. For is_cyclic=0 and a filter of size m_f, n_f the convolution uses only $m - m_f + 1 \times n - n_f + 1$ central pixels, and for is_cyclic=1 the convolution uses $n \times m$ pixels where the input is taken to wrap round as on a toroid.

   The output is of size $m \times n$. The optimization should be done in the spatial domain, solving a linear system of equations using Matlab's backslash. To build the convolution matrix A use the function **getConvMat.m** attached.

2. Frequency deconvolution **function est_x=deconvFreq(y,k,eta,sigma)**

   The same as above except that the deconvolution is solved in the frequency domain. The fiddly part in implementing frequency convolution and deconvolution in matlab is getting the array positioning and padding right so that the results are consistent with the conventional convolution. To help you do that, you are given the function **conv_fft2.m** which implements convolution in the frequency domain, the deconvolution should be implemented using a similar shitting and padding scheme.

2 Analysis

Use the file **data.mat** attached to this assignment. It contains a sharp image, 3 blur kernels and 3 images blurred with these kernels.

1. Use the clean image to estimate σ^2. A good estimate is *obsvar* where obsvar is the mean of the squared filter responses $(g_{ij}^x(x))^2$ and $(g_{ij}^y(x))^2$. You can also use the clean image to estimate the noise variance η^2. Include the estimated values in your report along with a brief explanation of how you obtained the noise estimators.

2. Deblur the blurry images using the two functions described above. Generate a (4×3) image array of the following form. Row 1 consists of the three blurred images. Row 2 shows the results of the primal deblurring with $is\_cyclic = 0$. Row 3 shows the results of the primal deblurring with $is\_cyclic = 1$. Row 4 shows the results of the frequency deblurring. Note that the results of rows 3 and 4 should be *equal* up to machine digitization errors, and we will take off points if this does not happen in your code.

   Use a variable named ImArray1 to store the image array.

   Save this image array as a .png image called ex1_q2.png, and add it to the report.

3. Generate a 3×3 image array where entry i, j is image $\#i$ deblurred with kernel $\#j$. Discuss the results in your report. You can use primal non cyclic deblurring here.

   Use a variable named ImArray2 to store the image array.

   Save this image array as a .png image called ex1_q3.png, and add it to the report.

4. Generate a 3×3 image array where entry i, j is image $\#i$ deblurred $\eta = 10\eta^*, \eta^*, 0.1\eta^*$ where η^* is the noise standard deviation you have estimated in 1. Discuss the results in your report. You can use primal non cyclic deblurring here.

   Use a variable named ImArray3 to store the image array.

   Save this image array as a .png image called ex1_q4.png, and add it to the report.

3 Submitted data

1. Two .m functions with the implementations of the primal and frequency deconvolutions.

2. One .m script named MainEx1.m which will go through all the subsections in part 2.

3. 3 output images requested in part 2.

4. A written PDF named studentID1_studentID2.pdf (write your ID numbers instead of studentID1, studentID2) report with the analysis requested in part 2 including the image array for each subsection.

5. Wrap all the matlab files, PDF and images into a single zip file name: studentID1_studentID2.zip (again use your IDs here).

4 Guidelines:

1. This is a Matlab programming assignment. It is very important that you thoroughly document your code. In particular each function should begin with a short description of what it does and what are the input and output.

2. Remember that Matlab is efficient when the code is vectorized, so try to avoid using loops. Specifically linear systems should be formulated in matrix notation and solved without using loops.

3. You will most probably want to use the following Matlab functions: help, imread, imwrite, imshow, figure, conv2, reshape, fft2, ifft2...

4. In this exercise you will have to use Matlab's sparse matrix representation. If you are not familiar with sparse matrices please consult the Matlab premier. The important thing to note here is that if you don't keep all your matrices sparse you will run out of memory.

5. Save each matlab function in a separate file.

6. A clarification regarding the image array, the variable should look as following:

$$ImArray1 = \begin{bmatrix} I_{1,1}^{m\times n} & I_{1,2}^{m\times n} & I_{1,3}^{m\times n} \\ I_{2,1}^{m\times n} & I_{2,2}^{m\times n} & I_{2,3}^{m\times n} \\ I_{3,1}^{m\times n} & I_{3,2}^{m\times n} & I_{3,3}^{m\times n} \\ I_{4,1}^{m\times n} & I_{4,2}^{m\times n} & I_{4,3}^{m\times n} \end{bmatrix}$$

In order to show the array use imshow(ImArray1).

5 FAQ:

1. The supplied function **getConvMat.m** has a flag that outputs a cyclic convolution or only the valid pixels. When you use it with the flag is_cyclic=0 it outputs zero rows for all rows corresponding to boundary pixels whose convolution is undefined. Note that you do not need to remove these rows manually. Since to solve the system you only use A^TA and A^Ty zero rows do not contribute to the output.

2. If y is $m \times n$, then A is $(n \cdot m) \times (n \cdot m)$. If your convolution matrix is not cyclic in practice your deconvolution problem is not using measurments (i.e. pixels in y) that are close to the boundary. In your reconstructed x some pixels very close to the boundary will appear blurry. This is fine and acceptable as part of the submission.

3. There is no need to zero pad y before computing its fourier transform. You can solve for an x the size of y (i.e. slightly smaller than the ground truth x provided).

4. To implement Fourier priors, you need to zero pad g_x g_y to get them to size $m \times n$ before computing their Fourier transform. Shifting them does not hurt but does not make any difference: shifting only changes the phase of the Fourier transform, and since the Wiener filter formula we have derived only involve $\|\mathcal{F}_{g^x}(\omega)\|^2$ phase does not change anything.

5. Said again: shifting and zero pedding before Fourier convolution and deconvolution is tricky. We have provided the function fft_conv to save you this mess. This function implements shifting and zero padding correctly, and you should just copy.

6. There is no need to build the matrix $\tilde{C}_k$ explicitly. You should just minimize the diagonal optimization problem we arrived at, also defined in (2).