UNIVERSITÄT TÜBINGEN
LEHRSTUHL FÜR COMPUTERGRAFIK
PROF. DR.-ING. HENDRIK P.A. LENSCH
JIEEN CHEN
05.07.2017

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

COMPUTATIONAL PHOTOGRAPHY
ASSIGNMENT 9

Submission deadline for the exercises: Thursday, 12. July 2017 before 11:59.

Instructions: Upload the source code to your solution **(no images please, just the plain code)** in the ILIAS system at:
ILIAS Computational Photography SoSe2018
The image in Figure 1(a) was convolved with the kernel shown in (b), resulting in the image in (c). In this image a considerable amount of Gaussian noise was added after blurring it to make the example more realistic. Your task is to deblur the blurred image using various deconvolution methods.

In the exercise, we padded the images at the boundaries to hide boundary artifacts. Then all images

(a) (b) (c)

Figure 1: From left to right: original image, blur kernel, blurred image.

are cropped in the end for display. This is provided in the code.

9.1 Fourier Deconvolution (30 points)

This assignment consists of deconvolving images with a known convolution kernel, which is known as non-blind deconvolution. First stop is naive Fourier deconvolution by division in the frequency domain. The image formation can be modeled as

$$i(x, y) = (p \otimes o)(x, y) + n(x, y) \tag{1}$$

where $i(x, y)$ is the captured image, $p(x, y)$ is the blur kernel, $o(x, y)$ is the latent image and $n(x, y)$ is iid Gaussian noise.
In the lecure you learned that a convolution equals to a elementwise multiplication in the frequency domain. This lead to a simple deconvolution scheme: Given the blurred image and the blur kernel you can compute the deconvolved original image as the following element-wise division in the Fourier domain:

$$\tilde{O}(\omega_x, \omega_v) = \frac{I(\omega_x, \omega_y)}{P(\omega_x, \omega_y)} \tag{2}$$

Your task is to implement this in Matlab using the Fast Fourier Transform (FFT) and its inverse. A sample result is shown in Figure 2.

(a) (b) (c)

Figure 2: Original image, blurred image and naive Fourier deconvolution result

Hints:

- For FFT you should use the Matlab function `fft2` and for inverse FFT you should use `ifft2`.
- For Fourier convolution, the FFT of the image and the FFT of the kernel should be the same size (otherwise you cannot apply the kernel in Fourier space).
- Be careful of the values you use to divide (remember - division by 0 is not defined, and division by very small numbers is numerically imprecise).

9.2 Fourier Deconvolution using Tikhonov Regularization (30 points)

From Exercise 7.1 we observe that with noise the naive approach performs not very well. The main problem stems from noise amplification due to the singularities when $P(w_x, w_y)$ approaches to zero. This can be solved by formulating an objective function with an image prior in form of a penalty term. In this assignment, the penalty term is a Tikhonov regularization on the gradient images of $o(x, y)$. The optimization objective becomes

$$J(o) = \|i(x, y) - (p \otimes o)(x, y)\|^2 + \lambda \|\nabla o(x, y)\|^2 \tag{3}$$

where λ is the regularization weight for the penalty term.
The Maximum-A-Posterior (MAP) estimate reads

$$\tilde{O}(\omega_x, \omega_y) = \frac{P^*(\omega_x, \omega_y) I(\omega_x, \omega_y)}{|P(\omega_x, \omega_y)|^2 + \lambda H(\omega_x, \omega_y)} \tag{4}$$

where $H(\omega_x, \omega_y)$ is the Fourier transform of the second order derivative(Laplacian) operator.
$P^*(\omega_x, \omega_y) = P(-\omega_x, -\omega_y)$ is the conjugate of $P(\omega_x, \omega_y)$, in MATLAB you can compute this with `conj`.
A sample result is shown in Figure 3 where $\lambda = 0.04$. In this assignment, your tasks are:

a) Implement the MAP estimate in Matlab.

b) Restoration quality will depend on the value λ. Compute results with different λ settings and use the GUI provided to explore their impact on the reconstruction.

(a) (b) (c)

Figure 3: Original image, blurred image and Tikhonov regularization result

Hints:

- $|P(\omega_x, \omega_y)|^2 = P^*(\omega_x, \omega_y) * P(\omega_x, \omega_y)$

9.3 Richardson-Lucy (RL) Deconvolution ($40 + 10^*$ points)

Your next assignment consists of deconvolving images with a given convolution kernel using the Richardson-Lucy algorithm (RL). The iterative algorithm can be formulated as

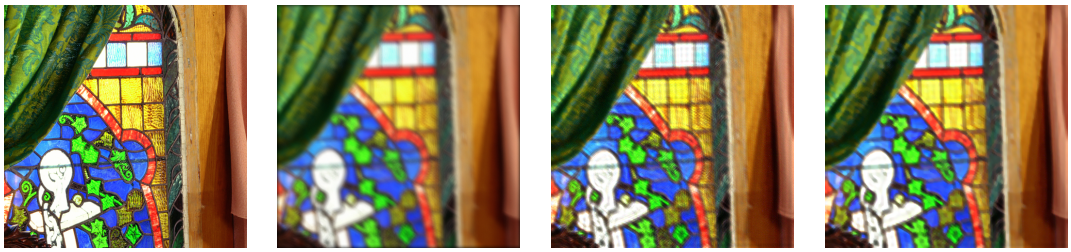
$$o^{n+1}(x, y) = o^n(x, y)\left[p^*(x, y) \otimes \frac{i(x, y)}{(o^n \otimes p)(x, y)}\right]\frac{1}{1 - \lambda\Gamma(x, y)} \tag{5}$$

where $o(x, y)$ is the estimated latent image, $i(x, y)$ is the blurred image, $p(x, y)$ is the blur kernel, $\Gamma(x, y)$ is the regularizer, λ is the regularization weight and $p^*(x, y) = p(-x, -y)$. Sample results are shown in Figure 4. Your tasks are:

a) Implement the RL algorithm (without regularizer, $\lambda = 0$). Compare the result with the Matlab built-in function `deconvlucy`.

(Bonus*) Extend the RL algorithm with a Total Variation(TV) regularizer. Please note that TV is implemented in the provided Matlab code.

b) Create a line plot PSNR versus λ.



(a) (b) (c) (d)

Figure 4: From left to right: original image, blurred image, deconvolved image without TV, deconvolved with $\lambda = 1.1$

Hints:

- You can use `conv2` function to compute two-dimensional convolution and `filter2` for correlation. Two-dimensional correlation is equivalent to two-dimensional convolution with the filter matrix rotated 180 degrees. For example in equation 6, you can use `conv2` to compute $(o^n \otimes p)(x, y)$ and `filter2` to compute $p^*(x, y) \otimes f(x, y)$, where $f(x, y) = \frac{i(x,y)}{(o^n \otimes p)(x,y)}$.