University of Michigan

EECS 504: Foundations of Computer Vision

Winter 2020. Instructor: Andrew Owens.

Problem Set 2 Solution: Signal Processing

Problem 2.1 *2D DFT and convolution theorem*

(a) Please match images on the left column of Figure 1 to its spectrum on the right. Please provide your answers in a colon separated format. For example, 1:A, 2:E, ... (1 point)

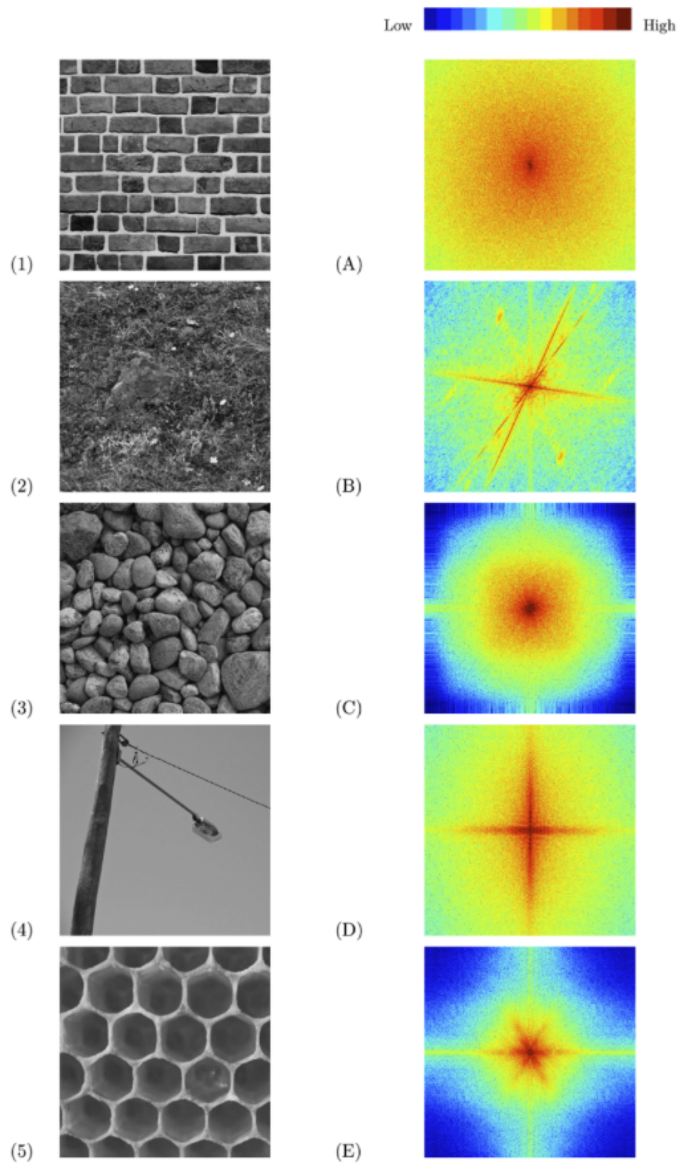

Figure 1: Images and DFT spectrum.

Answer:
(1) : (D), (2): (A), (3) : (C), (4) : (B), (5) : (E). (0.2 point each pair)

(b) Please prove that complex exponentials are eigenfunctions of linear shift-invariant systems, i.e., if we convolve a complex exponential $f[u] = \exp^{jwu}$ with a filter g, we get a complex scaled version $f * g = (a + bj)f$. Then, use this property to prove the *convolution theorem*, i.e., for two 2D signals $g, h \in \mathbb{R}^{M\times N}$,

$$f = g * h \Rightarrow F[u, v] = G[u, v]H[u, v] \tag{1}$$

where F, G, and H are the Fourier transforms of f, g and h respectively. In other words, the 2D DFT of the convolution of two signals is the product of their individual Fourier transforms. This property of 2D DFT is important because it allows us to perform linear filtering in the frequency domain by simple multiplication. (2 point)s

$$\begin{aligned}
f * g[u] &= \sum_v g[v]f[u-v], \quad f[u] = e^{j\omega u} \\
&= e^{j\omega u}\sum_v g[v]e^{-j\omega v} \\
&= e^{j\omega u}G(w) \quad \text{where } G(w) = \sum_v h[v]e^{-j\omega v} \\
&= f[u]G(\omega) \\
&= (a+bj)f[u] \quad \text{where } G(w) = a + bj
\end{aligned}$$

To prove the convolution theorem, we write the DFT of $f * g$ as

$$\begin{aligned}
\text{DFT}(f * g)[w] &= G(w)\sum_u f[u]\exp^{-jwu} \\
&= G(w)\cdot F[w]
\end{aligned}$$

(c) We look at a simple example to verify the convolution theorem on 2D images. In this problem, you will compare the output of a Gaussian filter, through i) direct convolution in the spatial domain and ii) product in the frequency domain. To perform DFT and inverse DFT, we use `fft2` and `ifft2` from `scipy.fft`. For 2D convolution, we use `scipy.signal.convolve2d`. (1 point)

Answer:
See the provided Colab notebook.

Problem 2.2 *Image blending*

Image pyramids are image representations useful for many downstream applications. This problem uses pyramid image processing to blend two images. We will use `OpenCV` to build image pyramids.

(a) As a first step, implement functions `pyr_up, pyr_down, gaussian_pyramid, laplacian_pyramid, reconstruct_img` to build a Laplacian pyramid from one image and show that you can reconstruct back the original image. Please use Gaussian kernels with the same size for `pyr_up and`

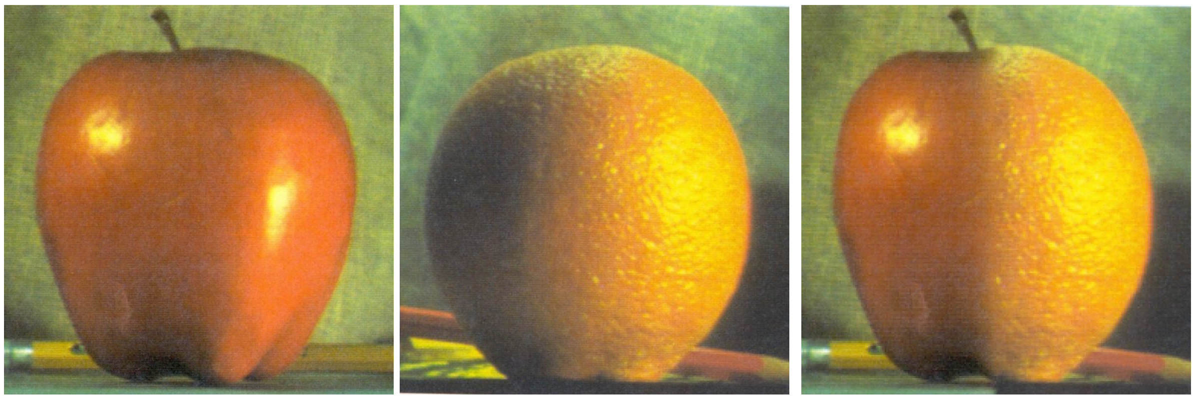

Figure 2: Blending with a Laplacian pyramid of 6 levels. Note that your result may look different from ours.

`pyr_down`. The only difference is that the kernel for `pyr_up` will be the one used for `pyr_down` multiplied by `4`. In your report, please include the original image, the Laplacian pyramid, and the reconstructed image. Please use a Laplacian pyramid with 4 levels. (Hint: `np.insert` may come in handy when implementing `pyr_up`) (4 points)

Answer:
See the provided Colab notebook.

(b) Implement functions `pyr_blend(im1, im2, mask, num_levels)` that takes as input two images and a binary mask (indicating which pixels to use from each image) and produces the Laplacian pyramids with `num_levels` levels for blending the two images. Use your function to blend the orange and apple image we provide in the Colab notebook. Include in your report the blended images with `num_levels` $\in \{1, 2, 3, 4, 5, 6\}$. Please describe the difference between the blended images with different levels of Laplacian pyramid. If you are asked by a company to pick the value of `num_levels` for an application with limited memory resources, what value will you choose? Why? (2 points)

Answer:
Pyramids with more levels consumes more memory during processing. For a memory limited application, we need make a good tradeoff between number of levels and the quality of the final image. Here we can choose `num_levels=3,4,5`. The reasoning is that increasing number of levels to 6 does not improve the output image quality significantly. (0.5 point)

For processed images, please see the provided Colab notebook. (1.5 point)