University of Michigan
EECS 504: Foundations of Computer Vision
Winter 2020. Instructor: Andrew Owens.

Problem Set 2: Signal Processing

Posted: Tuesday, January 21, 2020 **Due:** Tuesday, January 28, 2020

Please submit your written solution (all of Problem 2.1, except for (d)) to Gradescope as a `.pdf` file. The `.ipynb` file containing the visualizations we requested should be submitted to Canvas. Before submitting, please make sure to rename the file to `uniqname_umid.ipynb`.

The starter code can be found at:
https://drive.google.com/open?id=17RqJuoPdp4oHkSJIkHeS0rruSXRzsX2D

We recommend editing and running your code in Google Colab, although you are welcome to use your local machine instead.

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)

(b) Show that complex exponentials are eigenfunctions of linear shift-invariant systems, i.e., if we convolve a complex exponential $f[u] = e^{jwu}$ with a filter g, we get a complex scaled version: $f * g = (a + bj)f$. (2 point)

(c) (Optional) Use the above eigenfunction 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. (0 point)

(d) Convolve the provided image with a Gaussian filter: i) using direct convolution in the spatial domain, and ii) product in the frequency domain (via the convolution theorem). To perform DFT and inverse DFT, use `fft2` and `ifft2` from `scipy.fft`. For 2D convolution, we use `scipy.signal.convolve2d` (1 point).

(e) (Optional) How does the Fourier transform of the Gaussian filter in (d) change qualitatively as a function of σ? Provide an intuitive explanation for what's going on.

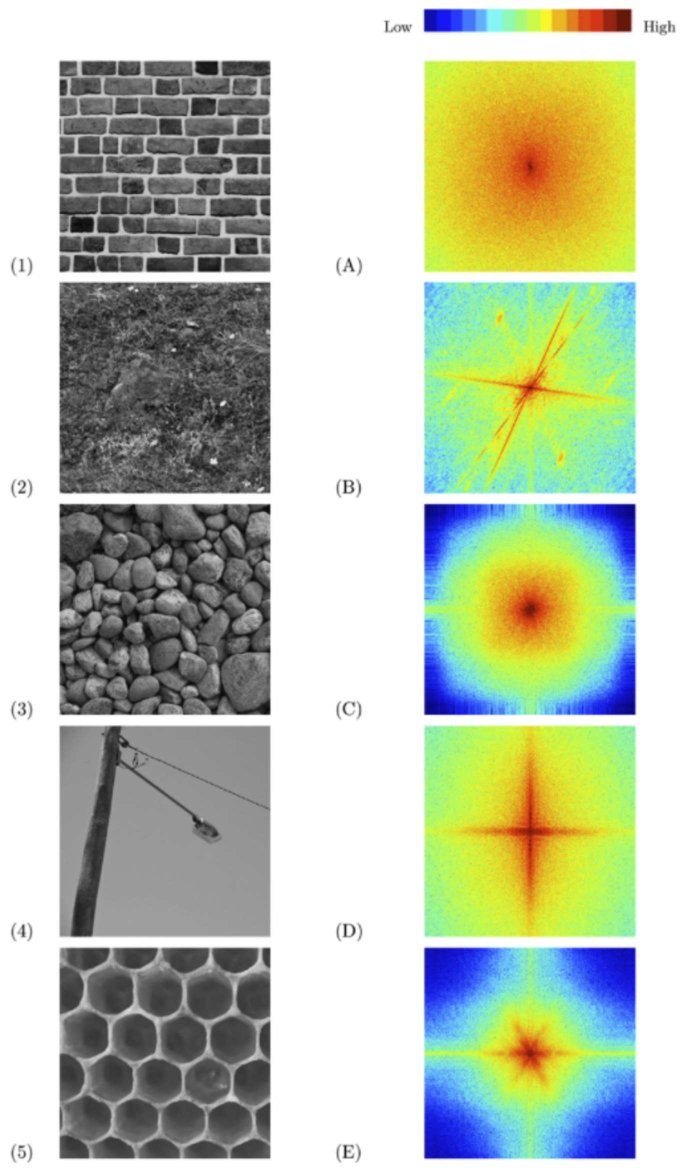

Figure 1: Images and DFT spectrum.

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.

(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 `pyr_down`. The only difference is that the kernel for `pyr_up` will be the one used for `pyr_down` multiplied by `4`. Please plot 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)

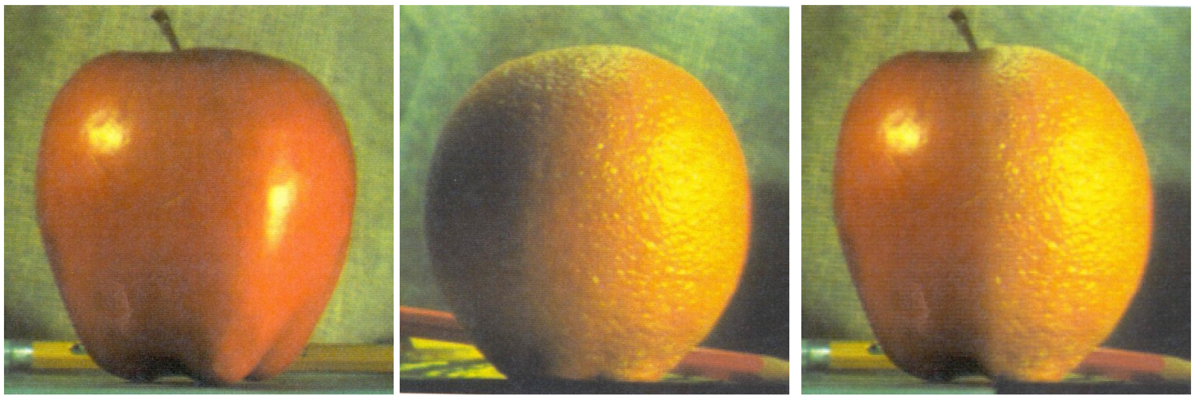

Figure 2: Blending with a Laplacian pyramid of 6 levels. Note that your result may look different than ours.

(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. Plot 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: how does the result change as you use more pyramid levels?

To obtain color images, you can apply the blending to each color channel independently. In our implementation, this did not require any extra code (the same code worked on single-channel and multi-channel images due to `numpy` broadcasting), but your implementation may differ. (2 points)

(c) (Optional) Use your code to blend your own images! If you would *not* like your blending results shown in class, please let us know!