ECE 285 – Assignment #5
Fourier transform

Written by Charles Deledalle on May 8, 2019.

In this assignment we will study the Fourier transform, and next implement convolutions in the Fourier domain as part of our image manipulation library `imagetools`.

First, start a Jupyter Notebook, go into the subdirectory `ece285_IVR_assignments` (or whatever you named it), and create a new notebook `assignment5_fourier.ipynb` with

```
%load_ext autoreload
%autoreload 2
import numpy as np
import numpy.fft as npf
import matplotlib
import matplotlib.pyplot as plt
import time
import imagetools.assignment5 as im

%matplotlib notebook
```

We will use

- `assets/house.png`
- `assets/map.png`
- `assets/motionblur.npy`
- `assets/montreuil.png`
- `assets/lady.png`

For the following questions, please write your code and answers directly in your notebook. Organize your notebook with headings, markdown and code cells (following the numbering of the questions).

1 Spectral analysis

1. In your notebook, create an image with a horizontal sinusoidal `a`:

   ```
   n = 256
   i = np.arange(n)
   j = np.pi / 16 * np.ones(n)
   x = np.sin(np.outer(i, j))
   ```

   Visualize this image and the amplitude of its spectrum in one single figure as

   ```
   fig, axes = plt.subplots(ncols=2, figsize=(7,3))
   im.show(x, ax=axes[0])
   im.showfft(x, ax=axes[1], apply_fft=True)
   fig.show()
   ```

Explain what you observe: number of bright points, their locations and their amplitude.

2. Repeat the same analysis for

   - a vertical sinusoidal:

     ```
     i2 = 2 * np.pi * 3 / 64 * np.ones(n)
     j2 = np.arange(n)
     x = 4 * np.sin(np.outer(i2, j2))
     ```

   - a variant of the vertical sinusoidal:

     ```
     i3 = 2 * np.pi * (3 + 1/8) / 64 * np.ones(256)
     x = np.sin(np.outer(i3, j2))
     ```

   - a diagonal sinusoidal

     ```
     x = np.sin(np.outer(i, j) + np.outer(i2, j2))
     ```

   - a square

     ```
     x = np.zeros((256, 256))
     x[62:190, 62:190] = 1
     ```

   - a sinusoidal in a window

     ```
     x = x * np.sin(np.outer(i, j))
     ```

3. Load the image `x = house`. Display this image and the logarithm of its spectrum as

   ```
   im.showfft(x, ax=axes[1], apply_fft=True, apply_log=True)
   ```

   What does the horizontal, vertical and oblique structures that you observe represent?

4. Repeat with the image `map`, `montreuil` and `lady`, and interpret what you observe.

5. For the three images, perform a spatial sub-sampling by a factor 2 in each direction

   ```
   x = x[::2, ::2]
   ```

   Inspect their spectrum, repeat with a factor 4, and explain how aliasing impacts these images.

2 Spectral convolution

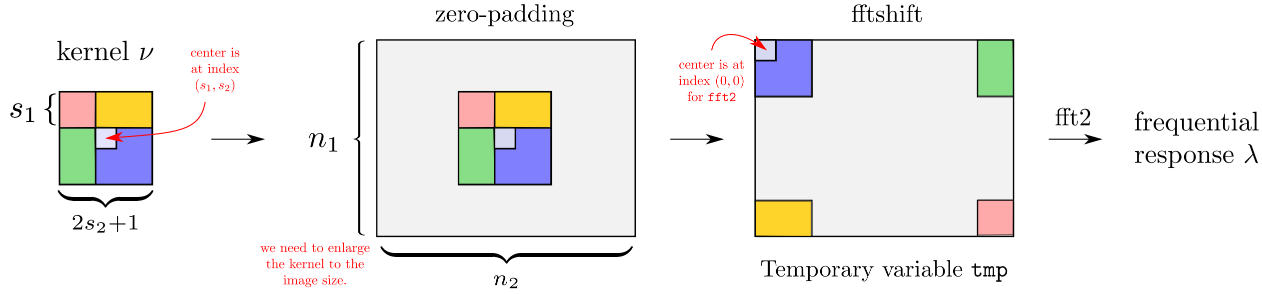
6. Run `convolve` on `x = lady` with a Gaussian convolution of bandwidth $\tau = 2$ using periodical boundary conditions. Display the image and its spectrum (in logarithm) before and after convolution. Explain what you observe.

7. Repeat with a box kernel $\tau = 5$. Explain what you observe.

8. Create a function

```python
def kernel2fft(nu, n1, n2, separable=None):
    ...
    tmp = np.zeros((n1, n2))
    tmp[:s1+1, :s2+1] = nu[s1:2*s1+1, s2:2*s2+1]
    ...
    return lbd
```

that creates a $n_1 \times n_2$ complex valued array `lambda` corresponding to the frequential response of the convolution kernel function with impulse response `nu` that is limited to window of support $[-s_1, s_1] \times [-s_2, s_2]$. The format of `nu` is the same as the one for `convolve` and is determined by the optional argument `separable` (default: `None`). Refer to the following figure for more details:



9. Create in `imagetools/assignment5.py`, the function

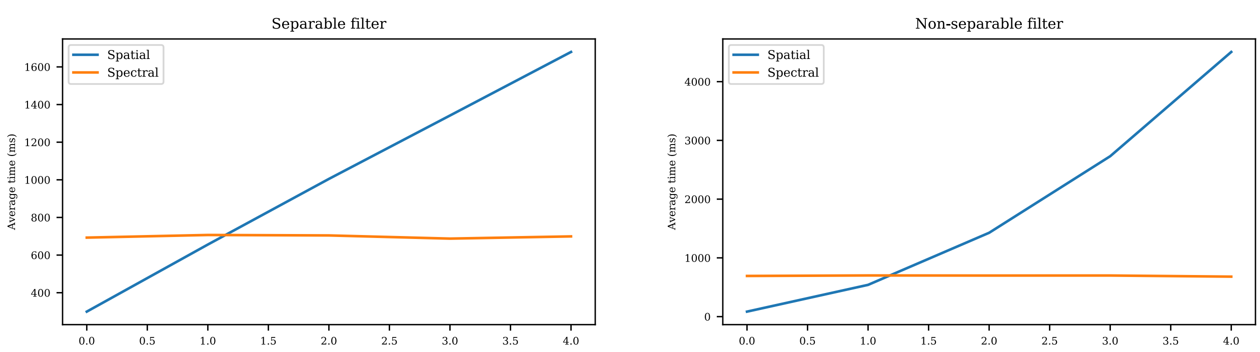
```python
def convolvefft(x, lbd)
```

that implements the convolution (in the Fourier domain) of x with a convolution kernel whose frequential response is given by λ.

Hint: Don't forget to take the real part.

10. Test your new convolution function on `x = lady`. Compare the results with `convolve` for different kernel functions using periodical boundary conditions. Check that the error between the two is zero (up to machine precision).

11. Compare the computation time of `convolve` and `convolvefft` for a box kernel with $\tau = 0$ to 4. Make the comparison when the spatial convolution is separable and non-separable (with the box kernel). Average the computation time on 100 runs. Draw the curves of average computation time as a function of τ (do not forget about title, axis names, and legend). Check that your curves are consistent with the ones below.



What are the complexities of the two methods with respect to τ and the separability? When is the spectral convolution favorable against the spatial convolution?

3 Adjoint

The convolution $\mathbf{H} : \mathbf{x} \mapsto \nu * \mathbf{x}$ where ν is the convolution kernel, is a linear operator with respect to x. As any linear operators, it has a unique adjoint $\mathbf{H}^*$ such that for all images x and y, $\langle \mathbf{H}x, y \rangle = \langle x, \mathbf{H}^* y \rangle$.

12. Copy paste the function `kernel` from `imagetools/assignment3.py` into `assignment4.py`. Modify the function in order to implement `name='motion'`, in which case it loads the convolution kernel stored in `assets/motionblur.npy`. All optional arguments are ignored in this case.

13. In your notebook, load the motion kernel, display it and display its application to `x = house` using `convolve`.

14. The adjoint $\mathbf{H}^*$ is also a convolution, $\mathbf{H}^* : \mathbf{x} \mapsto \mu * \mathbf{x}$. What is μ compared to ν? In your notebook, modify ν into μ. Using `convolve`, check that $\langle \mathbf{H}x, y \rangle = \langle x, \mathbf{H}^* y \rangle$ for `x = house` and `y = map`.

15. What is the frequential response of μ compared to the one of ν? In your notebook, using `kernel2fft`, create λ the frequential response of ν, and modify it into the one of μ. Using `convolvefft`, check that $\langle \mathbf{H}x, y \rangle = \langle x, \mathbf{H}^* y \rangle$ for `x = house` and `y = map`.