ECE 285 – Assignment #2

Writing part

---

1 Exercise 1 (Convolution)

Let f, g and h be functions defined from $\mathbb{Z}^2$ to $\mathbb{R}$. Recall that the convolution product reads

$$(f * g)(i,j) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} f(k,l)g(i-k, j-l), \quad \text{for all } (i,j) \in \mathbb{Z}^2 \ .$$

Show that

1. $f * g = g * f$.
2. $(f * g) * h = f * (g * h)$.

Practical part

---

2 Exercise 2 (Image shift)

Shifting an image x of size (n_1, n_2) in a direction (k, l) consists in creating a new image x^{shifted} of size (n_1, n_2) such that

$$x^{\text{shifted}}(i,j) = x(i+k, j+l) \ .$$

In practice, boundary conditions should be considered for pixels (i,j) such that $(i+k, j+l) \notin [0, n_1 - 1] \times [0, n_2 - 1]$. A typical example is to consider periodical boundary conditions such that

$$x^{\text{shifted}}(i,j) = x(i+k \mod n_1, j+l \mod n_2) \ .$$

1. Create in `imshift.m` a function implementing the shifting of an image x as

   ```
   function xshifted = imshift(x, k, l, boundary)
   ```

   where the fourth argument is a string that specifies the type of boundary conditions to use (refer to the class). It takes one of the values: `'periodical'`, `'extension'`, `'zero-padding'` or `'mirror'`.

   Hint: First write it using loops as

   ```
   case 'periodical'
       for i = 1:-k
           for j = 1:-l
               xshifted(i, j) = x(n1 + i + k, n2 + j + l);
       for ...
   ```

   Next try to get rid of the loops. Each case can be written in a few lines, as:

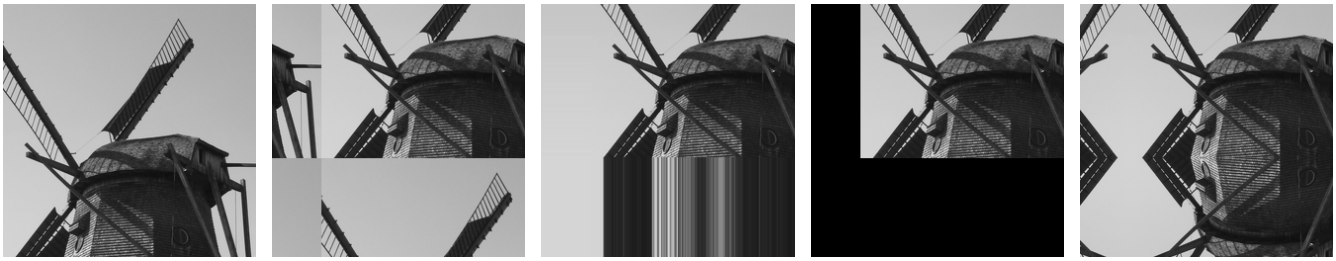
   ```
   case 'periodical'
       irange   = mod1((1:n1) + k - 1, n1) + 1;
       jrange   = mod1((1:n2) + l - 1, n2) + 1;
       xshifted = x(irange, jrange);
   ```

2. Download `assignment2.zip`, and extract the files

   (a) `assignment2/lake.png`

   (b) `assignment2/windmill.png`

   Write a script `test_imshift.m` that test your `imshift` function for the four boundary conditions on these images. The script should produce one figure with four subplots with the following results (do not forget to add a title to each subplot):



(a) Original (b) Periodical (c) Extension (d) Zero-padding (e) Mirror

Figure 1: Shift in the direction $(k, l) = (+100, -50)$ with different boundary conditions

3. Check on `x = windmill` and `y = lake`, if this operation is linear, i.e.,

```
imshift(a * x + b * y, k, l) == a * imshift(x, k, l) + b * imshift(y, k, l) ?
```

   Does that depends on the boundary conditions?

4. After shifting the image in the direction (k, l), shift it back in the direction $(-k, -l)$. Interpret the results. Which shift is one-to-one?

5. (Bonus) Depending on the boundary conditions: What is the kernel/null-space of this operation? What is the image/range of this operator? What is the adjoint? What is the pseudo-inverse?

   Hint: refer to the cookbook for definitions.

3 Exercise 3 (Convolution)

1. Create in `imkernel.m`, the function

```
function nu = imkernel(name, tau, s1, s2)
```

   that produces a function handle `nu` implementing a convolution kernel functions on the finite support $(-s_1, s_1) \times (-s_2, s_2)$. The first argument `name` is a string that specifies the type of kernel (refer to the class): `'gaussian'`, `'exponential'` or `'box'`. For the Gaussian case, the code reads as

```
case 'gaussian'
    w       = @(i, j) exp(-(i.^2 + j.^2) / (2 * tau^2));

    % normalization
    [i, j] = ndgrid(-s1:s1, -s2:s2);
    Z      = sum(sum(w(i, j)));
    nu     = @(i, j) w(i, j) / Z .* (abs(i) <= s1 & abs(j) <= s2);
```

2. Create in `imconvolve_naive.m`, the function

```
function xconv = imconvolve_naive(x, nu, s1, s2)
```

that performs (except around boundaries) the convolution between x and ν with four loops as

```
xconv = zeros(n1, n2);
for i = (s1+1):(n1-s1)
    for j = (s2+1):(n2-s2)
        for k = -s1:s1
            for l = -s2:s2
                % complete
```

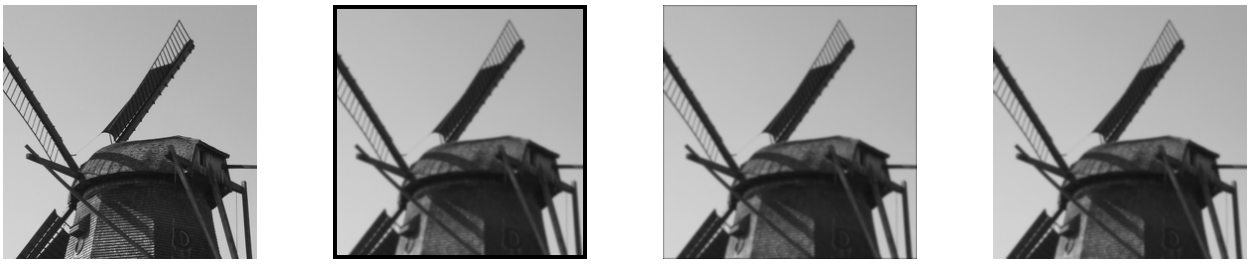
3. Create in `imconvolve_spatial.m`, the function

```
function xconv = imconvolve_spatial(x, nu, s1, s2, boundary)
```

that performs the convolution between x and ν including around boundaries. The idea is to switch the k, l loops with the i, j loops, and then make use of `imshift`. The final code should read with only two loops as

```
xconv = zeros(n1, n2);
for k = -s1:s1
    for l = -s2:s2
        xshift = imshift(x, -k, -l, boundary);
        % complete
```

4. Write a script `test_imconvolve.m` that compares the results and the execution times of `imconvolve_naive` and `imconvolve_spatial` for different boundary conditions (use `tic` and `toc`). Check that you obtain something similar to



(a) Original (b) Naive (12s) (c) Spatial+ZP (0.04s) (d) Spatial+mirror (0.03s)

Figure 2: Gaussian blur with $\tau = 1$ and $s_1 = s_2 = 4$.

5. Check on `x = windmill` and `y = lake`, if this operation is linear. Does that depends on ν? on the boundary conditions?

6. Choose `x = windmill` and `y = lake`, and compare the scalar products $\langle x, y^{\text{conv}} \rangle$ and $\langle x^{\text{conv}}, y \rangle$ for the Gaussian kernel and periodical boundary conditions. Here, we consider $\langle x, y \rangle = \sum_{i,j} x(i,j)y(i,j)$. What can you conclude about this convolution? Does this conclusion depends on ν? on the boundary conditions?