Homeworks for Sept. 29th

Simulation of motion blur 1D case  We wish to simulate what happens when the camera/observed scene moves during the exposure time. To this aim, we first consider here the simplest case: a 1 dimensional image, i.e., an image \( u \) with \( N \) columns but only one row. For the sake of simplicity we also assume a uniform relative motion between the camera and the observed scene and neglect the noise. This means that the observed scene at time \( t \) and pixel at position \( x \in \mathbb{R} \) is given by \( u(x - tv) \) where \( v \neq 0 \) is some constant. To simplify you can think that the camera is steady and that only the observed scene moves.

The goal is to design a simulator of motion blur. The inputs are 1) a discrete 1D image \( u \) (with positive pixel values), 2) \( v \neq 0 \) and 3) the exposure time \( \Delta t > 0 \). The output is just a motion blurred version of \( u \). You can directly give the pseudo code of this simulator if you can or you can first answer the following questions.

- Suppose that the exposure time \( \Delta t > 0 \) is given and that \( u \) is a continuous image, i.e., that you can observe \( u(x) \) for any \( x \in \mathbb{R} \). Give a formula for the observed value \( f \) a single pixel at position \( x \in \mathbb{R} \). Remember the scene \( u \) is moving during the exposure time.
- Re-write the above formula using a convolution.
- Deduce the pseudo-code of this simulator. You can assume w.l.o.g. that the discrete image \( u \) is not aliased. The inputs/output are given above.

Fourier transform of cardinal sines: ideal low-pass filter  1) Calculate\(^1\) the Fourier transform of \( f_K : \mathbb{R} \ni x \mapsto \frac{\sin(Kx)}{K} \), where \( K > 0 \) is a positive constant. We recall that \( \hat{f}(\xi) := \int_{\mathbb{R}} f(x) e^{-ix\xi} \, dx \).

2) Is the convolution \( u * f_K \) of a \([-\pi, \pi]\) band limited function \( u \) with \( f_K \) invertible for every \( K > 0 \) ? You shall briefly justify your answer. If applicable, you’ll give the interval (for \( K \)) so that this convolution is invertible.

3) Deduce the Fourier transform of \( \mathbb{R} \ni x \mapsto \frac{\sin(\pi x)}{\pi x} \). Briefly explain the name of ”ideal low pass filter” the convolution of a signal with a cardinal sine function.

4) Produce a code implementing an ideal low pass filter so that the image can be sub-sampled by a factor 2 in both directions. You’ll send an archive with the code and some comments.

A simple implementation of the bilateral filter  We’ll do the implementation of the bilateral filter in its simplest form. The bilateral filter produces its output from an weighted combination of neighboring pixels. The spatial and intensities kernels will be Gaussian functions with standard-deviation \( \sigma_s \) and \( \sigma_i \) respectively. The window defining the neighborhood will be a square window.

- The image is defined on the finite grid

\[
D = \{1, \ldots, N\} \times \{1, \ldots, M\} \subset \mathbb{N}^2.
\] (1)

- \( p = (p_1, p_2) \in D \) is the pixel to be processed
- \( u : D \rightarrow [0, 1] \) is the image, so \( u(p) \) is the value of the pixel \( p \)
- \( f_s(z) := \exp(-\frac{z^2}{2\sigma_s^2}) \), \( z \in \mathbb{R} \), is the Gaussian spatial kernel (it measures how far two pixels are), \( \sigma_s \) is a parameter
- \( f_i(z) := \exp(-\frac{z^2}{2\sigma_i^2}) \), \( z \in \mathbb{R} \), is the Gaussian intensity kernel (it measures how two gray-values differ), \( \sigma_i \) is a parameter

We want to denoise the pixel \( p \in D \), far enough from the boundaries of \( D \). To this aim we define:

- the square of size \((2w+1) \times (2w+1)\) (neighborhood) centered around \( p \) defined by

\[
\Omega(p) := p + \{-w, \ldots, w\}^2 = \left\{ y = (y_1, y_2) : y_1 \in \{p_1 - w, \ldots, p_1 + w\}, \; y_2 \in \{p_2 - w, p_2 + w\} \right\},
\] (2)

for \( p \) far enough from the boundaries of \( D \). The number \( w \in \mathbb{N}^+ \) is a parameter.

\(^1\)Calculate suggest that this is something to be done ”by hand” not using a computer.
The output of the bilateral filter is (we shall define the effective domain of \( p \) later on):

\[
\begin{align*}
\text{udenoised}(p) & := \frac{1}{C} \sum_{y\in\Omega(p)} u(y) f_s (\|y-p\|_2^2) f_i (|u(y) - u(p)|) \\
& = \frac{1}{C} \sum_{y_1=-p_1-w}^{p_1+w} \sum_{y_2=-p_2-w}^{p_2+w} u(y_1, y_2) \exp \left( -\frac{(y_1 - p_1)^2 + (y_2 - p_2)^2}{2\sigma_s^2} \right) \exp \left( -\frac{[u(y_1, y_2) - u(p_1, p_2)]^2}{2\sigma_i^2} \right),
\end{align*}
\]

where the normalization constant is given by

\[
C := \sum_{y\in\Omega(p)} f_s (\|y-p\|_2^2) f_i (|u(y) - u(p)|) \\
= \sum_{y_1=-p_1-w}^{p_1+w} \sum_{y_2=-p_2-w}^{p_2+w} \exp \left( -\frac{(y_1 - p_1)^2 + (y_2 - p_2)^2}{2\sigma_s^2} \right) \exp \left( -\frac{[u(y_1, y_2) - u(p_1, p_2)]^2}{2\sigma_i^2} \right).
\]

To denoise the entire image, loop over \( p \in D \) (and therefore \( \Omega(p) \)) so that \( \Omega(p) \subset D \) (as usual, be careful with the boundaries) and apply the above formulas.
Part 1. Before the implementation

Q1. Compute the range of $y - p$ when $y \in \Omega(p)$:

Q2. We recall that $s$ and $w$ are fixed. We consider the function $\Omega(p) \ni y \mapsto f_s(\|y - p\|_2)$. Compute the range of $f_s$. Hint: Use Q1.

Q3. Which part of $u$ do we need to compute $u_{\text{denoised}}(p)$?

Q4. From (3) compute the range of $p_1$ and $p_2$, i.e., the $p_1$ and $p_2$ for which (3) is valid

Q5. We recall that in Matlab, the first index of a vector is 1. Rewrite (3) so that the summation indexes starts at 1.
Q6. Rewrite the formula you obtained for Q5 using the function $S$ et $\tilde{u}$ given by

$$S : \{1, \ldots, 2w + 1\} \times \{1, \ldots, 2w + 1\} \ni (x_1, x_2) \mapsto \exp\left(-\frac{(x_1 - w - 1)^2 + (x_2 - w - 1)^2}{2\sigma^2}\right),$$  \hspace{1cm} (5)

$$\tilde{u} : \{1, \ldots, 2w + 1\} \times \{1, \ldots, 2w + 1\} \ni (x_1, x_2) \mapsto u(x_1 + p_1 - w - 1, x_2 + p_2 - w - 1).$$  \hspace{1cm} (6)

Q7. Deduce the similar formula for $C$ defined in (4).

Q8. List the objects needed to implement the program (Matrices of fixed size/varying size, matrices constant w.r.t the loop over $p$, real numbers, etc.)

Q9. Deduce the pseudo code of the program.
Part 2. Short analysis of the behavior of the bilateral filter

These questions are to answer before programming. Indeed, the goal is to be able to detect bugs or mistakes in the code. Not to describe the results!

Q10. What is the behavior of the bilateral filter when $\sigma_i \to +\infty$?

Q11. What happens when $\sigma_i \to 0$?

Q12. What happens when $\sigma_s \to 0$?

Q13. What happens when $\sigma_s \to +\infty$?
Part3. Programming  Complete the file bilateral.m and test your program. Write your program below.