Computational Photography Assignment 3

Single Michael
08-917-445

Task 1

Def(Linear) Given two measurable function f and g and α, $\beta \in \mathbb{C}$. An operator $\mathcal{F}$ is called *linear* if the following identity holds true:

$$\mathcal{F}\{\alpha f + \beta g\}(x) = \alpha \mathcal{F}\{f\}(x) + \beta \mathcal{F}\{g\}(x) \tag{1}$$

Def(Shift Invariant) Given a measurable function f. An operator $\mathcal{F}$ is called *shift invariant* if the following identity holds true:

$$\mathcal{F}\{S_\delta f\}(x) = S_\delta(\mathcal{F}\{f\})(x) \tag{2}$$

where the shift operator S_δ is defined as the following:

$$S_\delta(f)(x) = f(x + \delta) \tag{3}$$

For any given function f and small number $\delta \in \mathbb{R}$.

Note that notational convention the operation notion for any operator F, we write $F(f)(x)$ instead of $F(f(x))$. In words we apply the operator F (i.e. the functional F) to the function f. Thus is would even be possible to omit the argument x. However to make the reasoning clear, I will keep it.

Therefore, directly applying the definition from equation 3 to equation 2 we can also define the property *shift invariant* as the follwoing:

$$\mathcal{F}\left\{S_\delta(f(x))\right\} := \mathcal{F}\left\{f(x+\delta)\right\} = S_\delta(\mathcal{F}\left\{f\right\}(x) \tag{4}$$

Relying on the definitions of the equations before, equation 4 and equation 1 let I solved this task,

(a) $\mathcal{F}\left\{f\right\}(x) = e^{f(x)}$ is *not linear* and is *shift invariant*.

Show: $\mathcal{F}$ is not *linear*

Proof.

$$\begin{aligned}
\mathcal{F}\left\{\alpha f + \beta g\right\}(x) &= e^{\alpha f(x) + \beta g(x)} \\
&= e^{\alpha f(x)} e^{\beta g(x)} \\
&\neq \alpha e^{f(x)} + \beta e^{g(x)} \\
&= \alpha \mathcal{F}\left\{f\right\}(x) + \beta \mathcal{F}\left\{g\right\}(x)
\end{aligned}$$

□

Show $\mathcal{F}$ is *shift invariant*

Proof.

$$\begin{aligned}
\mathcal{F}\left\{S_\delta(f(x))\right\} &= \mathcal{F}\left\{f(x+\delta)\right\} \\
&= e^{f(x+\delta)} \\
&= S_\delta(e^{f(x)}) \\
&= S_\delta(\mathcal{F}\left\{f\right\}(x)
\end{aligned}$$

□

(b) $\mathcal{F}\left\{f\right\}(x) = f(x)f(x-1)$ is *not linear* and is *shift invariant*.

Show $\mathcal{F}$ is not *linear*

Proof.

$$\begin{aligned}\mathcal{F}\{\alpha f+\beta g\}(x) &= (\alpha f(x)+\beta g(x))(\alpha f(x-1)+\beta g(x-1))\\ &= \alpha^2 f(x)f(x-1)+\beta^2 g(x)g(x-1)+\alpha\beta(f(x)g(x-1)+f(x-1)g(x))\\ &\neq \alpha f(x)f(x-1)+\beta g(x)g(x-1)\\ &= \alpha\mathcal{F}\{f\}(x)+\beta\mathcal{F}\{g\}(x)\end{aligned}$$

□

Show $\mathcal{F}$ is *shift invariant*

Proof.

$$\begin{aligned}\mathcal{F}\{S_\delta(f(x))\} &= \mathcal{F}\{f(x+\delta)\}\\ &= f(x+\delta)f(x-1+\delta)\\ &= S_\delta(f(x)f(x-1))\\ &= S_\delta(\mathcal{F}\{f\}(x)\end{aligned}$$

□

(c) $\mathcal{F}\{f\}(x)=\sum_{k=x-4}^{x+2} f(k)$ is *linear* and is *shift invariant*.

Show $\mathcal{F}$ is *linear*

Proof.

$$\begin{aligned}\mathcal{F}\{\alpha f+\beta g\}(x) &= \sum_{k=x-4}^{x+2}(\alpha f(k)+\beta g(k))\\ &= \alpha\sum_{k=x-4}^{x+2} f(k)+\beta\sum_{k=x-4}^{x+2} g(k)\\ &= \alpha\mathcal{F}\{f\}(x)+\beta\mathcal{F}\{g\}(x)\end{aligned}$$

□

Show $\mathcal{F}$ is *shift invariant*

Proof.

$$\begin{aligned}\mathcal{F}\left\{S_\delta(f(x))\right\} &= \mathcal{F}\left\{f(x+\delta)\right\} \\ &= \sum_{k=x-4+\delta}^{x+2+\delta} f(k) \\ &= \sum_{k=x-4}^{x+2} f(k+\delta) \\ &= S_\delta(\sum_{k=x-4}^{x+2} f(k)) \\ &= S_\delta(\mathcal{F}\left\{f\right\}(x)\end{aligned}$$

□

(d) $\mathcal{F}\left\{f\right\}(x) = f(2x)$ is *linear* and is *not shift invariant*.

Show $\mathcal{F}$ is *linear*

Proof.

$$\begin{aligned}\mathcal{F}\left\{\alpha f + \beta g\right\}(x) &= \alpha f(2x) + \beta g(2x) \\ &= \alpha\mathcal{F}\left\{f\right\}(x) + \beta\mathcal{F}\left\{g\right\}(x)\end{aligned}$$

□

Show $\mathcal{F}$ is not *shift invariant*

Proof.

$$\begin{aligned}\mathcal{F}\left\{S_\delta(f(x))\right\} &= \mathcal{F}\left\{f(x+\delta)\right\} \\ &= f(2(x+\delta)) \\ &= f(2x+2\delta) \\ &\neq f(2x+\delta) \\ &= S_\delta(f(2x)) \\ &= S_\delta(\mathcal{F}\left\{f\right\}(x)\end{aligned}$$

□

(e) $\mathcal{F}\left\{f\right\}(x) = \sin(2x)f(x)$ is *linear* and is *not shift invariant*.

Show $\mathcal{F}$ is *linear*

Proof.

$$\begin{aligned}\mathcal{F}\left\{\alpha f+\beta g\right\}(x) &= \sin(2x)\left(\alpha f(x)+\beta g(x)\right)\\ &= \sin(2x)\alpha f(x)+\sin(2x)\beta g(x)\\ &= \alpha\sin(2x)f(x)+\beta\sin(2x)g(x)\\ &= \alpha\mathcal{F}\left\{f\right\}(x)+\beta\mathcal{F}\left\{g\right\}(x)\end{aligned}$$

□

Show $\mathcal{F}$ is not *shift invariant*

Proof.

$$\begin{aligned}\mathcal{F}\left\{S_\delta(f(x))\right\} &= \mathcal{F}\left\{f(x+\delta)\right\}\\ &= \sin(2x)f(x+\delta)\\ &\neq \sin(2x+\delta)f(x+\delta)\\ &= S_\delta(\sin(2x)f(x))\\ &= S_\delta(\mathcal{F}\left\{f\right\}(x)\end{aligned}$$

□

(f) $\mathcal{F}\left\{f\right\}(x)=xf(x)$ is *linear* and is *not shift invariant*.

Show $\mathcal{F}$ is *linear*

Proof.

$$\begin{aligned}\mathcal{F}\left\{\alpha f+\beta g\right\}(x) &= x(\alpha f(x)+\beta f(x))\\ &= x\alpha f(x)+x\beta f(x)\\ &= \alpha x f(x)+\beta x f(x)\\ &= \alpha\mathcal{F}\left\{f\right\}(x)+\beta\mathcal{F}\left\{g\right\}(x)\end{aligned}$$

□

Show $\mathcal{F}$ is not *shift invariant*

Proof.

$$\begin{aligned}\mathcal{F}\left\{S_\delta(f(x))\right\} &= \mathcal{F}\left\{f(x+\delta)\right\} \\ &= xf(x+\delta) \\ &\neq (x+\delta)f(x+\delta) \\ &= S_\delta(xf(x)) \\ &= S_\delta(\mathcal{F}\left\{f\right\}(x)\end{aligned}$$

□

(g) $\mathcal{F}\left\{f\right\}(x) = f(x) - f(x-5)$ is *linear* and is *shift invariant*.

Show $\mathcal{F}$ is *linear*

Proof.

$$\begin{aligned}\mathcal{F}\left\{\alpha f + \beta g\right\}(x) &= (\alpha f(x) + \beta g(x)) - (\alpha f(x-5) + \beta g(x-5)) \\ &= (\alpha f(x) - \alpha f(x-5)) + (\beta g(x) - \beta g(x-5)) \\ &= \alpha(f(x) - f(x-5)) + \beta(g(x) - g(x-5)) \\ &= \alpha\mathcal{F}\left\{f\right\}(x) + \beta\mathcal{F}\left\{g\right\}(x)\end{aligned}$$

□

Show $\mathcal{F}$ is *shift invariant*

Proof.

$$\begin{aligned}\mathcal{F}\left\{S_\delta(f(x))\right\} &= \mathcal{F}\left\{f(x+\delta)\right\} \\ &= f(x+\delta) - f(x-5+\delta) \\ &= S_\delta(f(x) - f(x-5)) \\ &= S_\delta(\mathcal{F}\left\{f\right\}(x)\end{aligned}$$

□

(h) $\mathcal{F}\left\{f\right\}(x) = f(x-3) - 2f(x-12)$ is *linear* and is *shift invariant*.

Show $\mathcal{F}$ is *linear*

Proof.

$$\begin{aligned}\mathcal{F}\{\alpha f + \beta g\}(x) &= (\alpha f(x-3) + \beta g(x-3)) - 2(\alpha f(x-12) + \beta g(x-12)) \\ &= (\alpha f(x-3) - 2\alpha f(x-12)) + (\beta g(x-3) - 2\beta g(x-12)) \\ &= \alpha(f(x-3) - 2f(x-12)) + \beta(g(x-3) - 2g(x-12)) \\ &= \alpha\mathcal{F}\{f\}(x) + \beta\mathcal{F}\{g\}(x)\end{aligned}$$

□

Show $\mathcal{F}$ is *shift invariant*

Proof.

$$\begin{aligned}\mathcal{F}\{S_\delta(f(x))\} &= \mathcal{F}\{f(x+\delta)\} \\ &= f(x-3+\delta) - 2f(x-12+\delta) \\ &= S_\delta(f(x-3) - 2f(x-12)) \\ &= S_\delta(\mathcal{F}\{f\}(x)\end{aligned}$$

□

Task 2

Given an $m \times n$ monochromatic (i.e. there is only one color-channel) Image I. Give an algorithm how to apply box-filtering on this image. Furthermore analyse the asymptotic complexity of this algorithm.

Algorithm 1 Moving Average box filter

Input: Grayscale *Image* I with resolution $m \times n$
Output: Box filtered Image *Image* $\hat{I}$

Procedures: $getDimensions(Image)$, $zeros(height, width)$

1: $[h, w] = getDimensions(I)$
2: $\hat{I} = zeros(h, w)$
3: $r = \lceil \frac{w-1}{2} \rceil$
4: **Foreach** $Pixel\, p \in Image\, I$ **do**
5: $\quad contribution = 0$
6: $\quad$ **Foreach** $Pixel\, p_n \in r - Neighborhood\ \mathcal{N}_r(p)$ **do**
7: $\quad\quad contribution = contribution + I(p + p_n)$
8: $\quad$ **end for**
9: $\quad \hat{I}(p) = \frac{contribution}{m \cdot n}$
10: **end for**

Remarks:

- By pixels in the Algorithm we are referring to the coordinates of the pixel in the image. Therefore p corresponds to the x and y coordinates of pixel p in the Image I.
- $I(p)$ denotes accessing the pixel-(color)-values in the images at the position of the pixel p in the image I.
- $\mathcal{N}_r(p)$ denotes the neighborhood with radius r around a given pixel p. In the context of pixel-coordinates, think of it as a box-grid, centred at the pixel coordinates of p. This grid has a radius of r. This means there are r neighbors (pixel-coordinates in the grid) below, on top, on the left and on the right of p.
- Our algorithm can easily be extended for color Images by simply applying the same algorithm to each color-channel separately.
- The assumption of being provided by a m by n can easily be extended for the case when $n \neq m$. This only will affect the computation of the radius r in algorithm .
  Computing $\lceil 0.5 \cdot \left(\lceil \frac{m-1}{2} \rceil + \lceil \frac{n-1}{2} \rceil \right) \rceil$ would be a valid option in order to compute r.
- If w (i.e. n) is odd, then $\lceil \frac{w-1}{2} \rceil$ is equal to $\frac{w-1}{2}$.
- The procedure *getDimensions* returns the width-and height resolution of a provided Image.
- The procedure *zeros* creates a new image with the provided resolutions.

Aysmptotic Complexity Next let us have a closer look into algorithm .

- The most outer foreach loop will iterate over each pixel. This loop corresponds to two for loops, one over iterating over each row in the image, another iterating over each column in the image. Assuming there are h rows and w columns in the image, this most outer foreach loop has an asymptotic complexity of $\mathcal{O}(h \cdot w)$. From the assignment description we are supposed to assume that there are m pixels in the image. Thus, we conclude, the most outer foreach loop has an asymptotic complexity in $\mathcal{O}(m)$.
- The inner foreach loop iterates over each neighorhood pixel around the current pixel in iteration (according the the most outer foreach loop). Assuming we are allowed to ignore boundary issue, we always visit a neighborhood of $\mathcal{O}\left((2r+1)^2\right)$ pixels. Referring to the definition of r at line 3 in algorithm we see that r linearly depends on the width of the image (note that we assume width is equal to height for the given image). Therefore $\mathcal{O}\left((2r+1)^2\right)$ is in the complexity class $\mathcal{O}\left(width^2\right)$.

We can conclude the following: The asymptotic complexity of algorithm is in $\mathcal{O}\left(k^2 m\right)$. Note that k denotes the width w from before and m is the number of pixels.

Task 3

Given a 3×3 box filter

$$B_{3\times 3} = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{5}$$

Applying the boxfilter from equation 5 twice corresponds to convolving the the box filter with itself, i.e. $B_{3\times 3} * B_{3\times 3}$. This will gives us the following filter:

$$B^2_{3\times 3} = \frac{1}{81}\begin{pmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{pmatrix} \tag{6}$$

In Matlab we can compute $B^2_{3\times3}$ applying the function *conv2* to the kernel $B_{3\times3}$ with itself, i.e.

$$B^2_{3\times3} = conv2(B_{3\times3}, B_{3\times3}) \tag{7}$$

Having a closer look to the filter $B^2_{3\times3}$ it resembles a gaussian bell (in 2d).

In spatial domain, the filter $B_{3\times3}$ is just a box function (here 2d). In the frequency domain, this corresponds to a *sinc* function. For simplification, let us consider the one-dimensional box function $rect_T(x)$ which is one if x is in the range $\left[-\frac{T}{2}, \frac{T}{2}\right]$ and zero otherwise.

$$rect_T(x) = \begin{cases} 1 & \text{if } -\frac{T}{2} \leq x \leq \frac{T}{2} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Next let us apply the Fourier Transformation on $rect_T(x)$:

$$\begin{aligned}
\int_{\mathbb{R}} rect_T(x) e^{-2j\pi\omega t} dt &= \int_{-\frac{T}{2}}^{\frac{T}{2}} e^{-2j\pi\omega t} dt && (9)\\
&= \frac{1}{2\pi i\omega} [e^{-2j\pi\omega t}]_{t=-\frac{T}{2}}^{\frac{T}{2}} && (10)\\
&= \frac{1}{2\pi i\omega} (e^{-j\pi\omega T} - e^{j\pi\omega T}) && (11)\\
&= \frac{T}{\pi\omega} \left(\frac{e^{j\pi\omega T} - e^{-j\pi\omega T}}{2j}\right) && (12)\\
&= \frac{T}{\pi\omega} (\sin(\pi\omega T)) && (13)\\
&= T sinc(\omega T) && (14)
\end{aligned}$$

From equation 14 we can conclude a box filter in the frequency domain is equal to a *sinc* function. Similarly, we can reason when applying a two dimensional box function. Using green's theorem, we can separate a multi-dimensional box function, when applying a Fourier operator (i.e. the n-dimensional Fourier Transformation), since a two dimensional box function is the cartesian product of two one dimensional rect functions. Notice that each of these rect functions only depends on one particular variable (dimension). Thus the integral from equation 14 is separable in the multidimensional case. Therefore, applying the box filter twice corresponds to $rect^2(x, y)$ which is equal to $sinc(u) * sinc(v)$ according to my previous reasoning. Thus, when applying the box function a infinite amount corresponds to the limit of *sinc* to the power of n, where n converges towards infinite. This converges towards a delta spike function, centred at zero (in the frequency domain). In the spatial domain, however, the function gets wider. Practically speaking, applying the box filter infinite times will average the whole image and the colors in the image will be everywhere all the same.

Task 4

In the following a list of steps that have to be performed in order to retrieve the 3×3 unsharp masking kernel:

1. Define the 3×3 Box Filter:

$$B_{3\times3} = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{15}$$

   Applying this filter will blur the image.

2. Define the 3×3 Identity Filter

$$I_{3\times3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{16}$$

   Applying this filter will give us the same image back.

3. The details of an image can be obtained by subtraction the blurred version of the image from itself. A kernel (filter) that does this for us can be obtained by subtracting the identity kernel from a blur kernel. For our case the 3×3 detail kernel $D_{3\times3}$ is equal to:

$$\begin{aligned} D_{3\times3} &= I_{3\times3} - B_{3\times3} && (17)\\ &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} && (18)\\ &= \frac{1}{9}\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} && (19) \end{aligned}$$

4. The detail of an Image can be scaled (enhanced or lowered) by multiplying a scalar factor λ to the kernel $D_{3\times3}$ when applying this kernel to an image.

5. Unsharp masking is simply scaling the details of an given image by applying a certain kernel. The kernel $D_{3\times3}$ from equation 19 will do the trick for us. The 3×3 unsharp masking kernel $U_{3\times3}$ can be obtained by adding a scalar multiple of the kernel $D_{3\times3}$ to the identity kernel $I_{3\times3}$.

$$U_{3\times3} = I_{3\times3} + \lambda D_{3\times3} \tag{20}$$

$$= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \frac{\lambda}{9} \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{21}$$

$$= \frac{\lambda}{9} \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{9}{\lambda} & 0 \\ 0 & 0 & 0 \end{pmatrix} + \frac{\lambda}{9} \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{22}$$

$$= \frac{\lambda}{9} \left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{9}{\lambda} & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \right) \tag{23}$$

$$= \frac{\lambda}{9} \begin{pmatrix} -1 & -1 & -1 \\ -1 & \frac{9}{\lambda} + 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{24}$$

Task 5

Task 6