Exercise Chapter 3 – Convolution

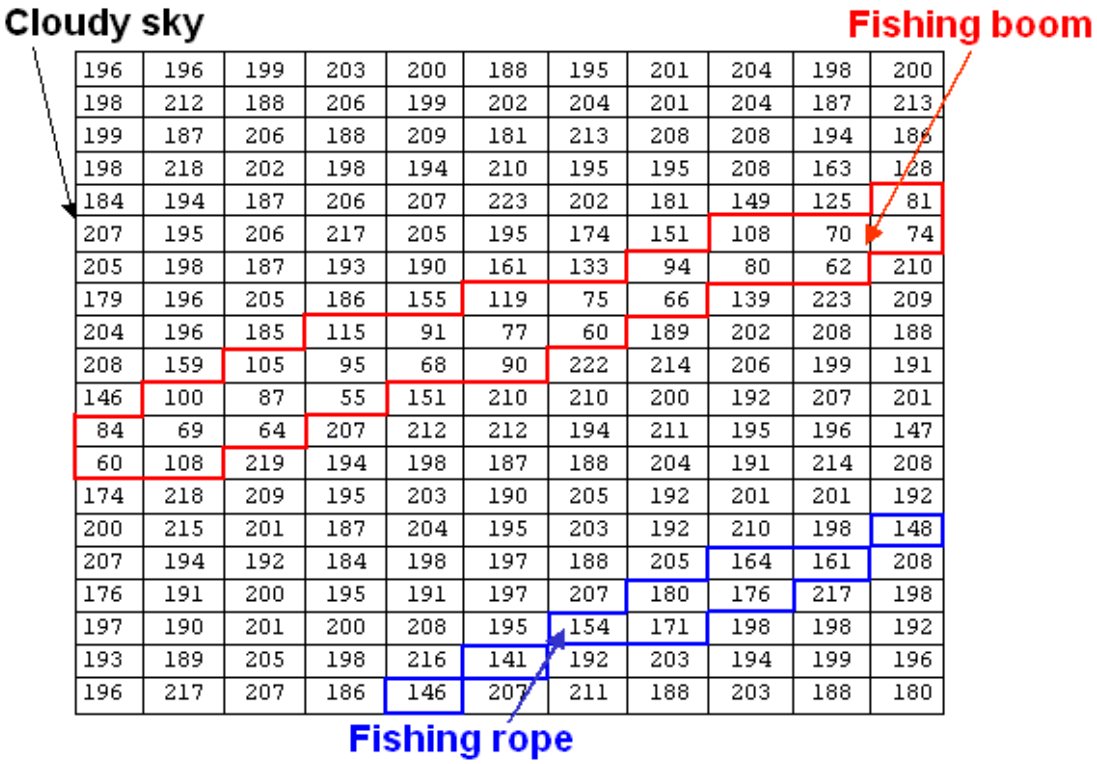
In this exercise, no programming is required. The goal is to perform a filtering by computing the convolution of an input image and the filter's kernel.

Convolution

We will use the convolution operator on a small portion of the original image "*BOATS*". On the figure below, this portion is delimited by a red rectangle that contains a part of the fishing boat's boom.

Image « BOATS »

The following array I_0 contains the luminance value of the pixels which are in the (20×11) delimited zone. In two diagonal bands, we can easily distinguish the pixels which belong to the fishing boom and to the rope.



196	196	199	203	200	188	195	201	204	198	200
198	212	188	206	199	202	204	201	204	187	213
199	187	206	188	209	181	213	208	208	194	186
198	218	202	198	194	210	195	195	208	163	128
184	194	187	206	207	223	202	181	149	125	81
207	195	206	217	205	195	174	151	108	70	74
205	198	187	193	190	161	133	94	80	62	210
179	196	205	186	155	119	75	66	139	223	209
204	196	185	115	91	77	60	189	202	208	188
208	159	105	95	68	90	222	214	206	199	191
146	100	87	55	151	210	210	200	192	207	201
84	69	64	207	212	212	194	211	195	196	147
60	108	219	194	198	187	188	204	191	214	208
174	218	209	195	203	190	205	192	201	201	192
200	215	201	187	204	195	203	192	210	198	148
207	194	192	184	198	197	188	205	164	161	208
176	191	200	195	191	197	207	180	176	217	198
197	190	201	200	208	195	154	171	198	198	192
193	189	205	198	216	141	192	203	194	199	196
196	217	207	186	146	207	211	188	203	188	180

The goal is to perform two kinds of filtering on some elements of this 2D array.

1 – Low-pass filtering

Let us consider a (3×3) low-pass filter. Its convolution kernel is:

$$\frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

After having performed this filter on the (20×11) area, we observe the following partial result:

111	149	149	150	149	146	148	150	150	149	113
149	199	198	199	199	198	201	203	201	197	149
149	200	199	198	198	199	202	204	200	188	135
148	200		198	201	203	200	196	185	161	105
146	197	199	203	206	205	194	175	151	121	75
149	197	199	203	201		169	142	112	96	75
148	197	196	193	181	157	129	108	102		113
145	195	189	170	146	117	98		140	175	146
146	185	163	131	105	96	115	155	187	200	149
134	154	121	98		126	169	197	202	200	147
99	111	95	109	143	178	201	204	200	196	143
68		117	158	189	200	201	200	199	193	138
78	130	169	193	199	196	196		199	198	145
123	185	200	199	196	195	196	197	200	198	143
150	204	200	194	196	197	197	196	194	187	134
148	197	194		194	196	196	191		184	141
142	192	195	195	196	194	189	183	184	193	149
142	193	197	201	197	186	179	182	191	198	148
146	197	200	198	189		184	191	195	195	144
112	153	151	141	133	139	147	148	147	144	105

Fill in the gray fields to have the full output array I_S (to round to the nearest integers less than or equal to the elements.).

2 – Contrast enhancement filtering

We wish to enhance the contrast of object edges in the delimited area. We use thus an "enhancement" filter which has the following convolution kernel:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

After having performed this filter, we observe the following partial result:

586	373	408	410	410	343	382	405	417	399	589
383	291	117	252	178	238	209	188	220	126	492
412	100	265	121	283	71	277	223	226	226	395
389	309	201	200		257	155	183	325	160	210
321	186	127	221	207	301	237	208	123	162	78
451	170	244	275	216	212	189	198	90		9
441	207	133	185	236	168	161	40	-3	-273	705
290	202	271	262	189	127		-167		497	424
437	236	304	18	40	25	-263	403	268	228	332
531	186		132		-127	536	253	223	183	367
338	39		-265	210	387	224	173	152	247	460
145	-11	-262	510	292	257	149	262		217	130
-66		520	151	194	147		238	141	274	487
392	384	212	182	228	160	252	158	211	200	403
404	262	202	151		181	235	150	295	270	142
465	165	181	148	214	207	128	301			533
285	195	221	200	157	195	316	141	121	352	373
426	172	210	198	238	275		120	251	184	368
383	140	230	183	387	-105	251	270	167	219	409
570	493	427	379	121	537	468	323	445	358	516

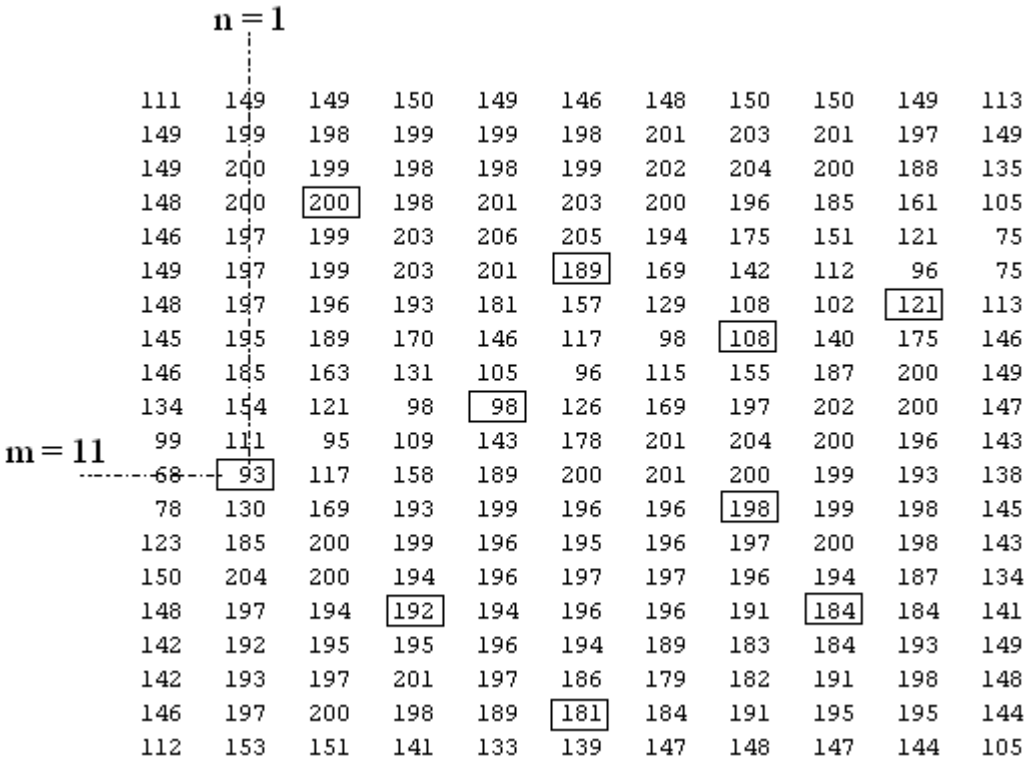
Again, fill in the gray fields to have the full output array.

3 – Border distortions

When computing an output pixel at the boundary of an image, a portion of the convolution kernel is usually off the edge of the image. Propose some examples of boundary conditions to fill in these “off-the-edge” image pixels.

Solution to the exercise on convolution

1 – Here is the 2D output array after low-pass filtering with the (3×3) binomial filter:



111	149	149	150	149	146	148	150	150	149	113
149	199	198	199	199	198	201	203	201	197	149
149	200	199	198	198	199	202	204	200	188	135
148	200	**200**	198	201	203	200	196	185	161	105
146	197	199	203	206	205	194	175	151	121	75
149	197	199	203	201	**189**	169	142	112	96	75
148	197	196	193	181	157	129	108	102	**121**	113
145	195	189	170	146	117	98	**108**	140	175	146
146	185	163	131	105	96	115	155	187	200	149
134	154	121	98	**98**	126	169	197	202	200	147
99	111	95	109	143	178	201	204	200	196	143
68	**93**	117	158	189	200	201	200	199	193	138
78	130	169	193	199	196	196	**198**	199	198	145
123	185	200	199	196	195	196	197	200	198	143
150	204	200	194	196	197	197	196	194	187	134
148	197	194	**192**	194	196	196	191	**184**	184	141
142	192	195	195	196	194	189	183	184	193	149
142	193	197	201	197	186	179	182	191	198	148
146	197	200	198	189	**181**	184	191	195	195	144
112	153	151	141	133	139	147	148	147	144	105

Here are all the steps to compute the output value of the pixel $I_S(11, 1)$:

$$I_S(11,1) = \frac{1}{16} \cdot \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} h(i,j) \cdot I_0(11-i, 1-j)$$

$$= \frac{1}{16} \cdot \sum_{i=-1}^{+1} h(i,-1) \cdot I_0(11-i, 1-(-1)) + h(i,0) \cdot I_0(11-i, 1-0) + h(i,1) \cdot I_0(11-i, 1-1)$$

$$= \frac{1}{16} \cdot \Big[h(-1,-1) \cdot I_0(11+1, 2) + h(-1,0) \cdot I_0(11+1, 1) + h(-1,1) \cdot I_0(11+1, 0)$$

$$+ h(0,-1) \cdot I_0(11, 2) + h(0,0) \cdot I_0(11, 1) + h(0,1) \cdot I_0(11, 0)$$

$$+ h(1,-1) \cdot I_0(12-1, 2) + h(1,0) \cdot I_0(12-1, 1) + h(1,1) \cdot I_0(12-1, 0) \Big]$$

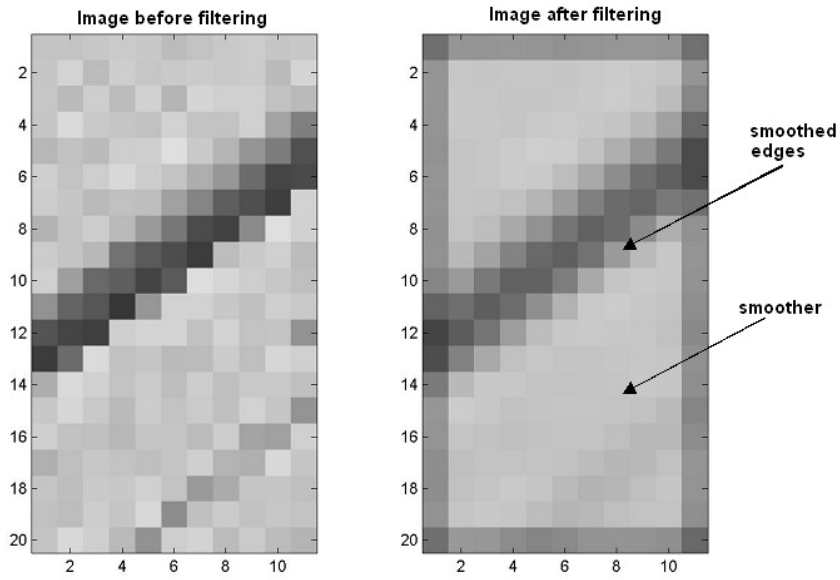
$$= \frac{1}{16} \cdot [\, 1x219 + 2x108 + 1x60 + 2x64 + 4x69 + 2x84 + 1x87$$

$$+ 2x100 + 1x146 \,]$$

$$= 1500/16 = 93.75$$

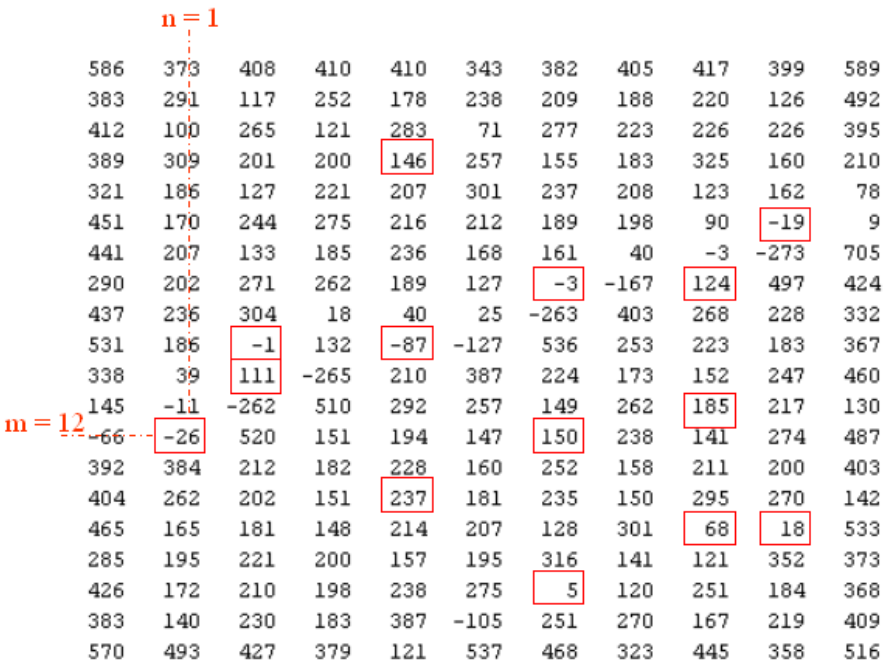
Thus: $I_S(11,1) = E[\, 93.75 \,] = 93$

Displayed here are the images before (on the left) and after (on the right) filtering:



Here, the smoothing effects (caused by removing the high spatial frequencies) of the low-pass filter are strongly visible.

2 – Here is the array after performing the 3×3 enhancement filter:



Here are all the steps to compute the output value of the pixel $I_S(12, 1)$ for example:

$$Is(12,1)=\sum_{i=-1}^{+1}\sum_{j=-1}^{+1} h(i,j).I_0(12-i,1-j)$$

$$=\sum_{i=-1}^{+1} h(i,-1).I_0(12-i,1-(-1))+h(i,0).I_0(12-i,1-0)+h(i,1).I_0(12-i,1-1)$$

$$=\Big[h(-1,-1).I_0(12+1,2)+h(-1,0).I_0(12+1,1)+h(-1,1).I_0(12+1,0)$$

$$+h(0,-1).I_0(12,2)+h(0,0).I_0(12,1)+h(0,1).I_0(12,0)$$

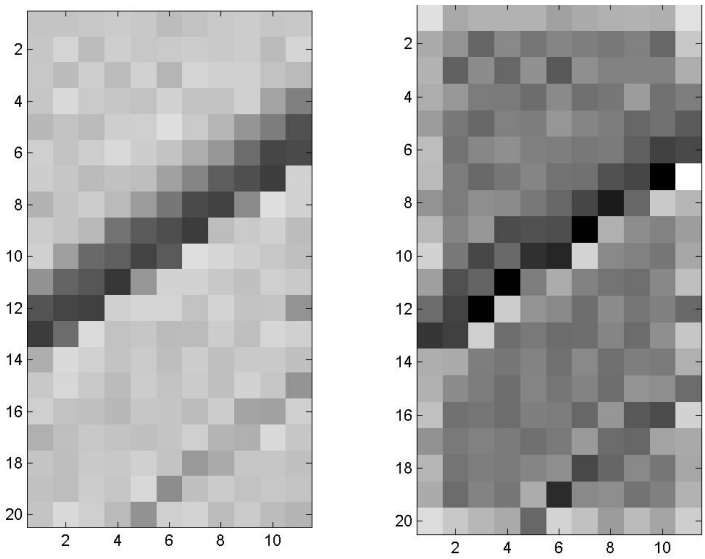
$$+h(1,-1).I_0(12-1,2)+h(1,0).I_0(12-1,1)+h(1,1).I_0(12-1,0)\Big]$$

$$= 0x209 + (-1)x218 + 0x174 + (-1)x219 + 5x108 + (-1)x60 + 0x64 + (-1)x69 + 0x84$$

$$= -26$$

Note: here, the output luminance values can be negative or higher than 255. To display the image result, it is thus necessary to scale the gray levels to the full range [0, 255].

Here is the display of the arrays before (on the left) and after (on the right) filtering:



The enhancement filter which is performed is indeed the sum of an Identity filter and a Laplacian filter (useful for edge detection):

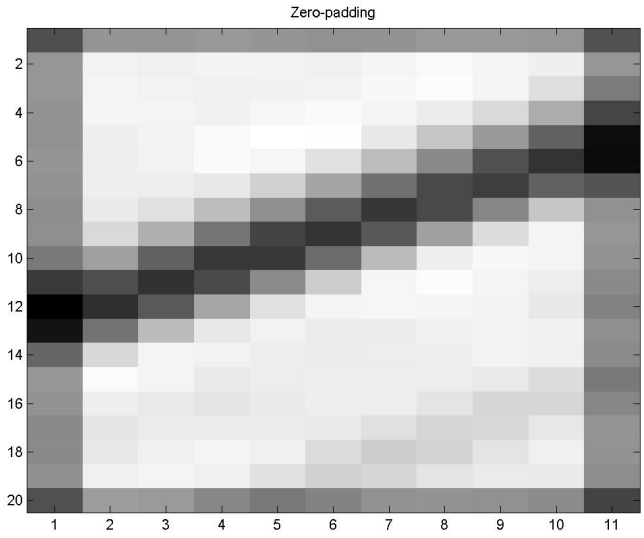
$$\underset{\text{Identity}}{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}} + \underset{\text{Laplacian}}{\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}} \equiv \underset{\text{Enhancement}}{\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}}$$

This filter enhances the contrast of the objects (like the fishing boom and rope) in the original image.

3 – A lot of methods are defined to compensate the border distortions: zero-padding, border replication, etc.
For more information about these boundary conditions, refer to the correction of the exercise: “Filtering” (chapter3).
Here, we will only show some results for the low-pass filtering with two methods used to remove these border distortions.

- Zero-padding:

That is the simplest method: the "off-the-edge" neighborhood is set to 0. Zero-padding can result in a dark band around the edge of the image:



- Replication:

The "off-the-edge" neighborhood is set to the value of the nearest border pixel in the original image (to filter using border replication, pass the optional argument ‘replicate’ to the Matlab function ***imfilter***). The border replication eliminates the zero-padding artifacts around the edge of the image.