exam_lectures_answers

November 12, 2024

Image Processing (IMAJS) – 2024-2025

2024/10/07 – Jean-Christophe Taveau – 1h00 - Documents allowed

The images are defined at the end of this page.

1 Digital image

1.1 Input Data

0	1	8	6
2	2	1	1
1	15	14	12
3	6	9	10

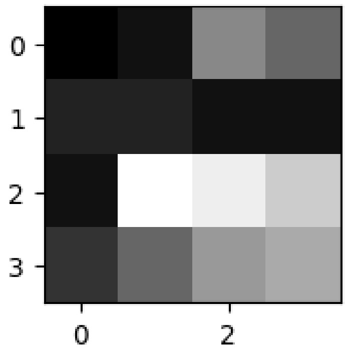
Fig.1: Image

[1]:

```python
import numpy as np

fig1 = [
    [0,  1,  8,  6],
    [2,  2,  1,  1],
    [1, 15, 14, 12],
    [3,  6,  9, 10]
]

fig1 = np.array(fig1)
```

[41]:

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(3,3))
plt.imshow(fig1,cmap='gray')
plt.show()
```



1.2 Questions

1.2.1 Question 1.1. What is the size of the image of Fig. 1?

Answer: width x height = 4 x 4

[15]:

```
print('height x width:', fig1.shape)
```

```
height x width: (4, 4)
```

1.2.2 Question 1.2. What is the minimal number of bits required for encoding the image of Fig. 1

(1-bit, 2-bit, 3-bit, ..., 8-bit, 9-bit, 10-bit, ..., 16-bit)? Justify.

Answer: 4 bits because pix values are in the range of 0-15. $2^4 = 16$ *gray levels*.

[16]:

```
np.min(fig1),np.max(fig1),2**4
```

[16]:

```
(np.int64(0), np.int64(15), 16)
```

[18]:

```
# Using a `while` loop
i = 2
while 2**i <= np.max(fig1):
    i = i + 1
print(f'{i}-bit image')
```

```
4-bit image
```

[19]:

```
# Using a `for .. in` loop
for i in range(2,16):
    if 2**i > np.max(fig1):
        print(f'{i}-bit image')
        break;
```

```
4-bit image
```

1.2.3 Question 1.3. Convert the image of Fig. 1 into hexadecimal notation.

Answer

```
0 1 8 6
2 2 1 1
1 F E C
3 6 9 A
```

[6]:

```python
fig1_hex = []
for p in fig1.flatten():
    fig1_hex.append(hex(p))

fig1_hex = np.array(fig1_hex).reshape(fig1.shape)
fig1_hex
```

[6]:

```
array([['0x0', '0x1', '0x8', '0x6'],
       ['0x2', '0x2', '0x1', '0x1'],
       ['0x1', '0xf', '0xe', '0xc'],
       ['0x3', '0x6', '0x9', '0xa']], dtype='<U3')
```

1.2.4 Question 1.4. What is the value of the pixel of coordinates (3,2) in image of Fig. 1?

Answer

P(3,2) = 12

[7]:

```python
x = 3
y = 2
fig1[y,x]
```

[7]:

```
np.int64(12)
```

1.2.5 Question 1.5. What are the XY-coordinates of the pixel at index 13 in image of Fig. 1?

Give the intermediate calculations.

```
x = index % width = 13 % 4 = 1
```

```
y = index // width = 13 // 4 = 3
```

[21]:

```python
index = 13
width = 4
x = index % width
y = index // width
print('x =',x,'; y =',y)
```

```
x = 1 ; y = 3
```

1.2.6 Question 1.6. Extract all the one-bit channels from the image of Fig. 1.

This image may be considered as a color-like image composed of N one-bit channels where N is the number of bits required to encode this image. A one-bit channel is encoded as 1 bit.

Answer

The image is encoded in 4 bits. Thus, the decimal pixel values are in binary:

10->2	10->2	10->2	10->2	10->2	10->2	10->2	10->2
0 = 0000	1 = 0001	2 = 0010	3 = 0011	4 = 0100	5 = 0101	6 = 0110	7 = 0111
8 = 1000	9 = 1001	10=1010	11=1011	12=1100	13=1101	14=1110	15=1111

Thus, the `fig1` in binary becomes:

```
[0000][0001][1000][0110]
[0010][0010][0001][0001]
[0001][1111][1110][1100]
[0011][0110][1001][1010]
```

Finally, we extract each bit and gathered them in each plane.

```
bit=3   bit=2  bit=1   bit=0
0010    0001   0001    0100
0000    0000   1100    0011
0111    0111   0110    1100
0011    0100   1101    1010
```

[9]:

```python
# use of bit mask
print('bit=3\n', (fig1 & 0b1000 == 8) * 1)
print('bit=2\n', (fig1 & 0b0100 == 4) * 1)
print('bit=1\n', (fig1 & 0b0010 == 2) * 1)
print('bit=0\n', (fig1 & 0b0001 == 1) * 1)
```

```
bit=3
 [[0 0 1 0]
 [0 0 0 0]
 [0 1 1 1]
 [0 0 1 1]]
bit=2
 [[0 0 0 1]
 [0 0 0 0]
 [0 1 1 1]
 [0 1 0 0]]
bit=1
 [[0 0 0 1]
 [1 1 0 0]
 [0 1 1 0]
 [1 1 0 1]]
bit=0
```

```
[[0 1 0 0]
 [0 0 1 1]
 [1 1 0 0]
 [1 0 1 0]]
```

1.2.7 Question 1.7. Write the formula to normalize the image of Fig. 1 between 0 and 64?

Write the formula

min = 0 and max = 15

p’ = (p - 0) / (15 - 0) * 64 = p / 15 * 64   p * 4

Draw the expected histogram? L’histogramme est le suivant:

bins	0	1	2	3	6	8	9	10	12	14	15
norm	0	4	8	12	25	34	38	42	51	59	64
count	1	4	2	1	2	1	1	1	1	1	1

[42]:

```
import matplotlib.pyplot as plt

fig1_norm = np.floor(fig1 / 15 * 64)
print(fig1_norm)
```

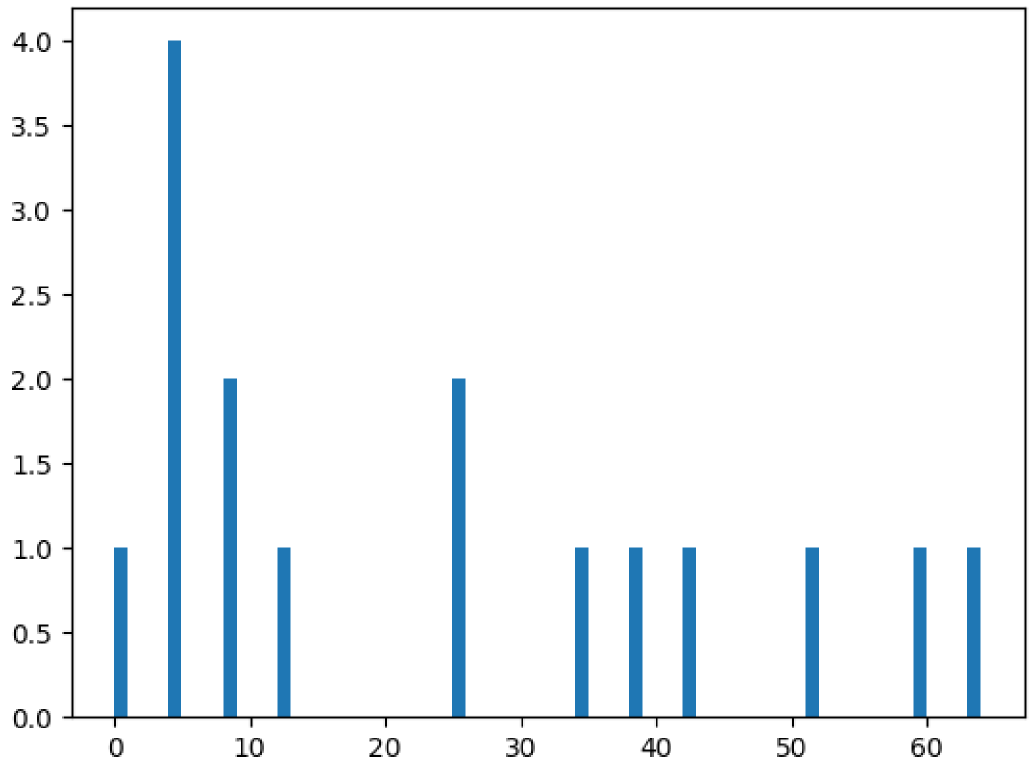
```
[[ 0.  4. 34. 25.]
 [ 8.  8.  4.  4.]
 [ 4. 64. 59. 51.]
 [12. 25. 38. 42.]]
```

[43]:

```
plt.hist(fig1_norm.flatten(),bins=64)
```

[43]:

```
(array([1., 0., 0., 0., 4., 0., 0., 0., 2., 0., 0., 0., 1., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 2., 0., 0., 0., 0., 0., 0., 0., 0.,
        1., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,
        1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1.]),
 array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11., 12.,
        13., 14., 15., 16., 17., 18., 19., 20., 21., 22., 23., 24., 25.,
        26., 27., 28., 29., 30., 31., 32., 33., 34., 35., 36., 37., 38.,
        39., 40., 41., 42., 43., 44., 45., 46., 47., 48., 49., 50., 51.,
        52., 53., 54., 55., 56., 57., 58., 59., 60., 61., 62., 63., 64.]),
 <BarContainer object of 64 artists>)
```



2 Image Processing

2.1 Input Data

```
0  0  0  1  1  0
0  1  1  1  1  1
0  1  1  1  1  1
0  1  1  1  1  1
0  0  1  1  1  0
0  0  0  1  0  0
```

Fig.2: 0=False; 1=True

```
0  0  0  0
1  1  0  0
0  1  1  1
0  0  0  1
```

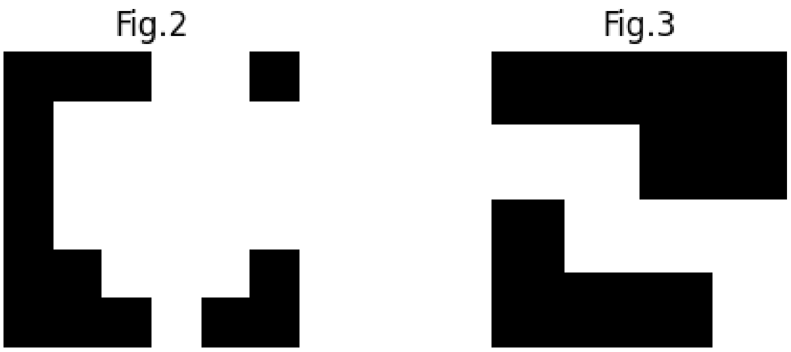
Fig.3: 0=False; 1=True

[64]:
```
fig2 = np.array([
    [0,  0,  0,  1,  1,  0],
    [0,  1,  1,  1,  1,  1],
    [0,  1,  1,  1,  1,  1],
    [0,  1,  1,  1,  1,  1],
    [0,  0,  1,  1,  1,  0],
    [0,  0,  0,  1,  0,  0]
])
```

```
fig3 = [
    [0,  0,  0,  0],
    [1,  1,  0,  0],
    [0,  1,  1,  1],
    [0,  0,  0 , 1]
]
```

[71]:

```
fig,ax = plt.subplots(1,2, figsize=(6,2))
ax[0].imshow(fig2,cmap='gray')
ax[0].axis('off')
ax[0].set_title('Fig.2')
ax[1].imshow(fig3,cmap='gray')
ax[1].axis('off')
ax[1].set_title('Fig.3')
plt.show()
```



2.2 Questions

2.2.1 Question 2.1. Apply a mean filter 3x3 to the pixel of coordinates (2,2) in image of Fig. 1.

Answer: (2 + 1 + 1 + 15 + 14 + 12 + 6 + 9 + 10) / 9 = 7.7   8

[26]:

```
(2 + 1 + 1 + 15 + 14 + 12 + 6 + 9 + 10 ) / 9
```

[26]:

```
7.777777777777778
```

In Python,

[11]:

```
#
subset = fig1[1:,1:]
print(subset)
np.mean(subset)
```

```
[[ 2  1  1]
 [15 14 12]
 [ 6  9 10]]
```

```
[11]: np.float64(7.777777777777778)
```

2.2.2 Question 2.2. Apply a median filter 3x3 to the pixel of coordinates (2,2) in image of Fig. 1

Answer: $1 < 1 < 2 < 6 < [\mathbf{9}] < 10 < 12 < 14 < 15$

```
[12]: print('Sort',np.sort(subset.flatten()))
      np.median(subset)
```

```
Sort [ 1  1  2  6  9 10 12 14 15]
```

```
[12]: np.float64(9.0)
```

2.2.3 Question 2.3. If we apply a Gaussian filter 3x3 to the image of Fig. 1 without padding the image, what will be the size of the output (filtered) image? Justify.

width x height = 2 x 2 because the kernel size is 3 x 3 reducing the output image of one pixel all along the edges.

2.2.4 Question 2.4. What is the result of thresholding the image of Fig. 1 at a value of 7?

Give the algorithm in pseudo-code and write the thresholded image.

```
for each pixel do:
    if pixel >= threshold:
        output_pixel = True
    else:
        output_pixel = False
    endif
endfor
```

Answer: For sake of convenience, 0 = False; 1 = True.

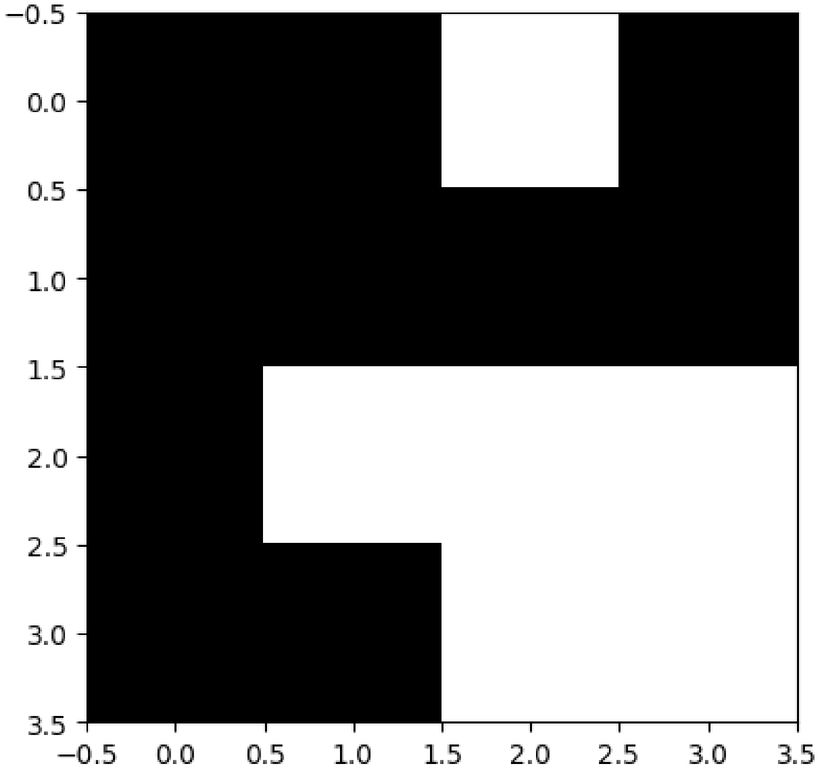
```
0010
0000
0111
0011
```

```
[13]: # numpy array
      fig1 >= 7
```

```
[13]: array([[False, False,  True, False],
             [False, False, False, False],
             [False,  True,  True,  True],
       [False, False,  True,  True]])
```

```
[29]: plt.imshow(fig1 >= 7,cmap='gray')
```

```
[29]: <matplotlib.image.AxesImage at 0x7f2e3ba4a120>
```



2.2.5 Question 2.5. From the previous binary image, apply an erosion.

Give the pseudo-code and write the eroded image.

Pseudo-code

```
for each subset_3x3:
    if True in subset_3x3:
        central_pixel = False
```

Answer

> **Note**: For the padding, the pixels outside the image are considered as `True`. This is an easy way to *neutralize* them.

```
0 = False; 1= True
0000
```

```
0000
0000
0001
```

```
[23]: import skimage as ski

      kernel = np.ones((3,3))
      ski.morphology.erosion(fig1 >= 7,kernel)
```

```
[23]: array([[False, False, False, False],
             [False, False, False, False],
             [False, False, False, False],
             [False, False, False,  True]])
```

2.2.6 Question 2.6. Calculate the euclidean distance map of the image of Fig. 2. If any, give the coordinates of the UEPs.

```
0  0  0  1  1  0
0  1  1  1  1  1
0  1  2  2  2  2
0  1  1  2  1  1
0  0  1  1  1  0
0  0  0  1  0  0
```

No UEPs.

```
[40]: erosion1 = ski.morphology.erosion(fig2,kernel)
      erosion2 = ski.morphology.erosion(erosion1,kernel)
      erosion3 = ski.morphology.erosion(erosion2,kernel)
      fig2 + erosion1 + erosion2 + erosion3
```

```
[40]: array([[0, 0, 0, 1, 1, 0],
             [0, 1, 1, 1, 1, 1],
             [0, 1, 2, 2, 2, 2],
             [0, 1, 1, 2, 1, 1],
             [0, 0, 1, 1, 1, 0],
             [0, 0, 0, 1, 0, 0]])
```

```
[45]: import scipy.ndimage as spi

      spi.distance_transform_cdt(fig2)
```

```
[45]: array([[0, 0, 0, 1, 1, 0],
             [0, 1, 1, 1, 1, 1],
             [0, 1, 2, 2, 2, 2],
             [0, 1, 1, 2, 1, 1],
             [0, 0, 1, 1, 1, 0],
             [0, 0, 0, 1, 0, 0]], dtype=int32)
```

2.2.7 Question 2.7. What is the euclidean distance between the points P1(0,1) and P2 (3,3)? Give the formula and the intermediate calculations.

[25]:

```
import math

d = math.sqrt( (0 - 3)**2 + (1 - 3)**2)
print(d)
```

```
3.605551275463989
```

2.2.8 Question 2.8. After image processing, the image of Fig. 1 is converted into the image of Fig. 3 .

```
0  0  0  0
1  1  0  0
0  1  1  1
0  0  0  1
```

What is the measured distance between these two points (in bold)? Justify. The simplest way is to count the number of True pixels. Here is 6 pixels. It is a 4-pixel connectivity.

> **Note**: If you subtract one pixel. It is correct too.

How is it possible to get the correct distance? Justify.

> Because it is a 4-pixel connectivity, we have to divide the 6 / 1.273 = 4.7 which must be compared to euclidean distance of 3.6

[ ]:

```

```