☐0 ☐0
☐1 ☐1
☐2 ☐2
☐3 ☐3
☐4 ☐4
☐5 ☐5
☐6 ☐6
☐7 ☐7
☐8 ☐8
☐9 ☐9

Image and Signal Processing

M1 MoSIG
December 16, 2013

Duration: 3 hours
Lecture Notes and Computer Exercises allowed
Calculators allowed

Exercises 1 and 2 are independent.

Exercise 1: Mean and Median Filters *(10 points)*

1.1 Mean Filter

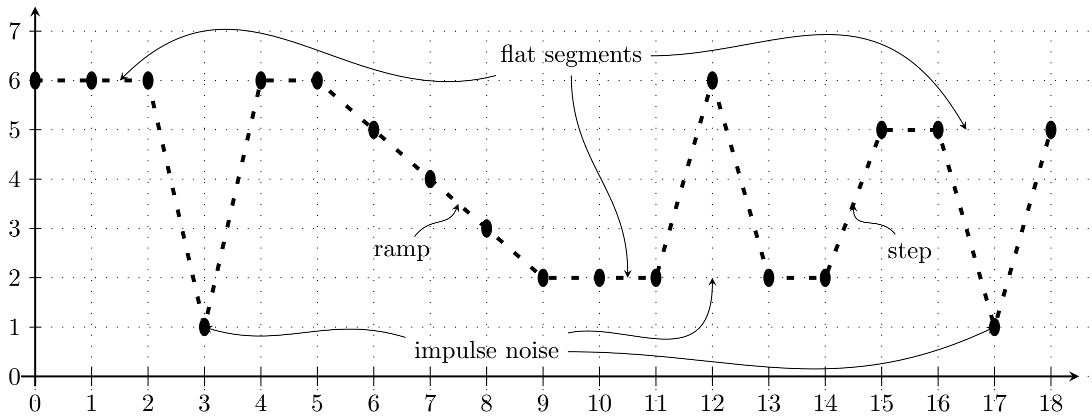
Question 1 What is a *Mean Filter* ? How does it work ? How is it implemented (in the spatial domain) ? Give an example of 2D 3×3 Mean Filter kernel.

☐A ☐B ☐C ☐D ☐E *For examiner only (do not check)*

Question 2 Complete Figure 1 (a) (graphics) and (b) (numbers) with the output signal of a *Mean Filter* of width 3 applied to the proposed signal.

.... ☐**A** ☐**B** ☐**C** ☐**D** ☐**E** *For examiner only (do not check)*

(a) *Represented as a 1D signal and its* Mean *and* Median *filters outputs*



6	6	6	1	6	6	5	4	3	2	2	2	6	2	2	5	5	1	5

(b) 3×3 *Mean Filter* (Question 2)

X												3						X

(c) 3×3 *Median Filter* (Question 5)

X												2						X

Figure 1: Mean and Median filters of width 3 applied on a 1D signal.

In the spatial domain, if the noise to remove can be modelled as a Gaussian, a better filter exists to remove it.

Question 3 What is the name of this filter ? How does it work ?

☐**A** ☐**B** ☐**C** ☐**D** ☐**E** *For examiner only (do not check)*

1.2 Median Filter

Some noises however cannot be modeled as Gaussian. One of them, named *Impulse noise* or *Salt & Pepper* adds some uncorrelated values anywhere in the signal/image. This noise affects one sample/pixel independently of its neighbors.

To filter it, we commonly use a *Median Filter*. In a *Median Filter* of width w, each input value/pixel is replaced by the median of the pixels contained in a surrounding window. This can be expressed by

$$x[n] \longrightarrow \boxed{\text{Median Filter}} \longrightarrow y[n]$$

$$y[n] = median\left\{x[n+i], i \in -\frac{w}{2} < i < \frac{w}{2}\right\}$$

This means that for each signal sample $x[n]$, a *Median Filter* of width $w = 3$ looks at $x[n-1], x[n]$ and $x[n+1]$, order them in increasing order and takes as a final value for $y[n]$, the *median* value, that is to say the middle value once input values have been ordered.

<u>Example:</u> $x[n] = \{1, 5, 2, 4, 3\}$

To compute $y[2]$, we look at

n	1	2	3
$x[n]$	5	2	4

and order them in increasing order $\boxed{2 \mid 4 \mid 5}$ the output value $y[2]$ is then the middle value 4, which is the *median value* of $\boxed{5 \mid 2 \mid 4}$.

In the same way, $y[0] = X$ (undefined), $y[1] = median\{1, 5, 2\} = 2$, $y[2] = median\{5, 4, 2\} = 4$, $y[3] = median\{2, 4, 3\} = 3$, $y[4] = X$ (undefined).

So we have: $y[n] = \{X, 2, 4, 3, X\}$.

Question 4 Is this filter linear ? Prove it.

What are the outputs y_1 and y_2 of the *Median Filter* on the 2 following signals ?

- $x_1 = \{1, 2, 3, 4, 5\}$ and
- $x_2 = \{0, 3, -1, 0, -2\}$

☐**A** ☐**B** ☐**C** ☐**D** ☐**E** *For examiner only (do not check)*

Question 5 Complete Figure 1 (a) (graphics) and (c) (numbers) with the output signal of a *Median Filters* of width 3 applied to the proposed signal.

.... A B C D E *For examiner only (do not check)*

Question 6 Compare the results with the *Mean Filter*. What do you conclude ?

A B C D E *For examiner only (do not check)*

Question 7 A 5×5 Mean and Median filters have been applied to the following image (where salt and pepper noise can be observed):

Which of the following image is obtained with the *Median Filter* ?

☐ ☐

Exercise 2: Signal Compression *(10 points)*

2.1 Image Example

In the following, a 2D image would be considered as a signal $I[k]$, $k \in [0..63]$, represented in 255 gray levels $[0, 1, ...255]$ such that $\forall k \in [0, ..15], I[k] \in [0, ..255]$.

Figure 2 represents a gray level image (at full resolution on the right and subsampled for the sake of the exercise on the left) with values presented on the left. *Even though in this sample, only a part of gray level are represented, the image may contain gray levels from* 0 *to* 255.

250	250	250	254	255	254	250	255
250	140	150	255	250	250	150	230
255	250	180	150	255	150	150	250
255	254	255	180	140	200	255	255
255	255	255	150	180	200	254	255
255	255	180	230	255	200	250	254
255	200	230	254	255	255	150	255
250	250	254	255	255	254	250	255

Figure 2: Gray leven image with normal display on the left and gray levels display on the right.

Definition 0.1 (Histogram) *The* **histogram** $H = \{h_i, i \in [1..L]\}$ *of an image* $I = \{I[k], k \in [0..N-1]\}$ *displays the number of samples there are in the signal that have each possible values.*

$$\forall i \in [1, L] h_i = card\{k \in [0, N-1]/I[k] = i\}$$

Definition 0.2 (Normalized Histogram) *The propability distribution of the image corresponds to the normalized hitogram* $P = \{p_i, i \in [1..L]\}$ *such that*

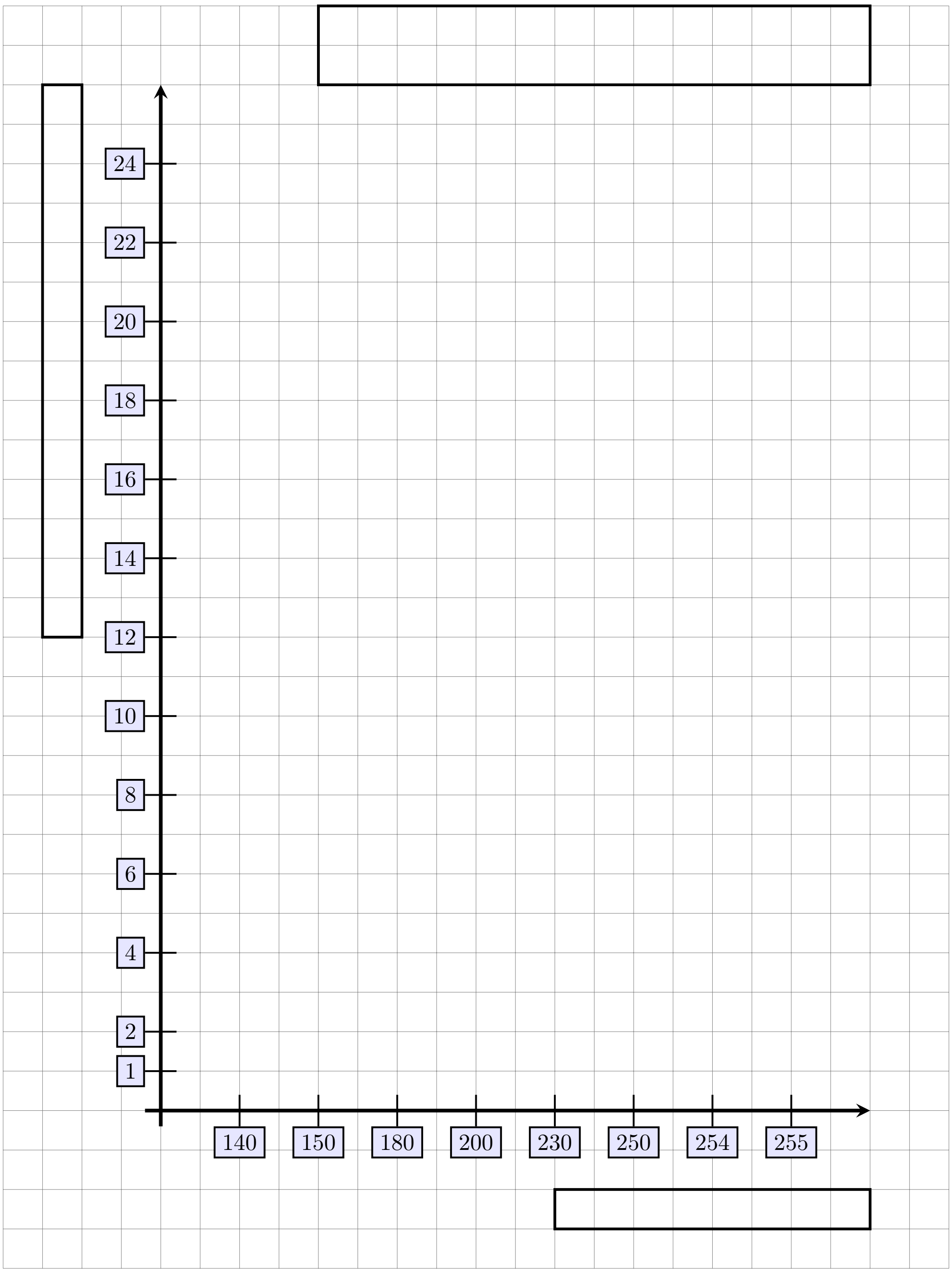
$$\forall i \in [1, L] p_i = \frac{h_i}{N}$$

Question 8 Complete the following table for the image shown in Figure 2.

gray levels	0	...	140	150	180	200	230	250	254	255
h_i	0	0	2	7	4		3	13	8	23
p_i			0.031			0.063				

.... A B C D E *For examiner only (do not check)*

Question 9 Complete the following histogram for image shown in Figure 2.



.... A B C D E *For examiner only (do not check)*

Question 10 On how many bits (binary digits) would be encoded one gray level of the image of Figure 2 ?

.... ☐A ☐E *For examiner only (do not check)*

Question 11 What would be the final size (in *bits*) of the whole image of Figure 2 ?

.... ☐A ☐E *For examiner only (do not check)*

2.2 Huffman encoding

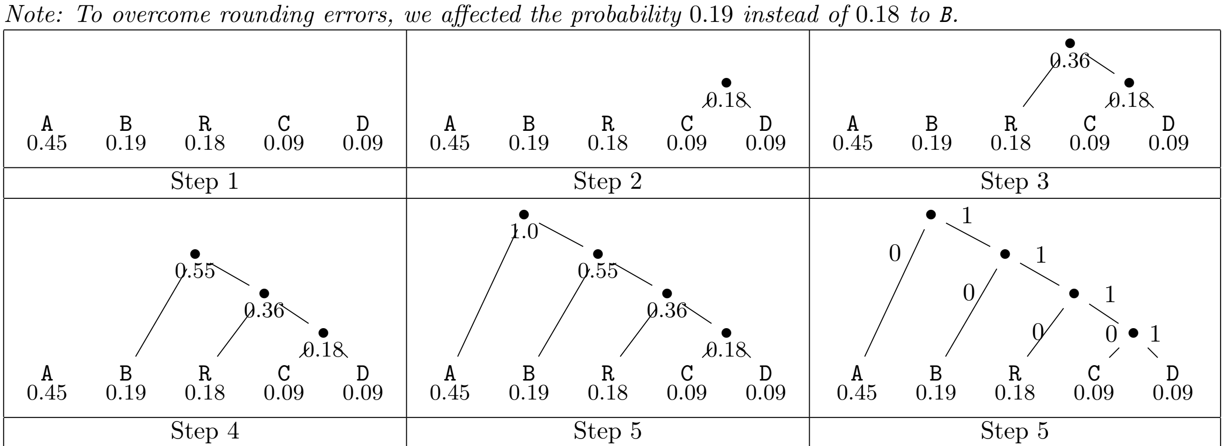
To compress without loss (i.e. without loosing information), we can use Huffman encoding. This encoding, instead of taking a fixed length to code the value of each pixel, creates a binary code where the code for a gray level which often appears in the image is shorter (i.e. takes less storage place) than the code of a gray level which almost never appears in the image.

To understand how Huffman encoding works, let us considere an alphabetic example: the word `ABRACADABRA`. In this word, the letter `A` appears 5 times (i.e. with a probability $p_A = 0.45$), the letters `B` and `R` appear 2 times each (i.e. with a probability $p_B = p_R = 0.18$), the letters `C` and `D` appear 1 time each (i.e. with a probability $p_C = p_D = 0.09$). Thus the letter `A` will have a shorter code than `D`.

To build the code, we build a binary tree.

- Step 1: Order samples (letters) according to their probabilities.
- Step 2: Link the 2 nodes (leaves here) with the smallest probabilities and calculate the cumulated probability
- Steps 3, 4, 5: continue linking 2 nodes with the smallest probabilities
- Step 6: Affect the code 0 to left branches of the final tree and 1 to the right branches of the same tree.
- Step 7: Summarize the obtained encoding.

Note: To overcome rounding errors, we affected the probability 0.19 *instead of* 0.18 *to* `B`.



A	B	C	D	R
0	10	1110	1111	110

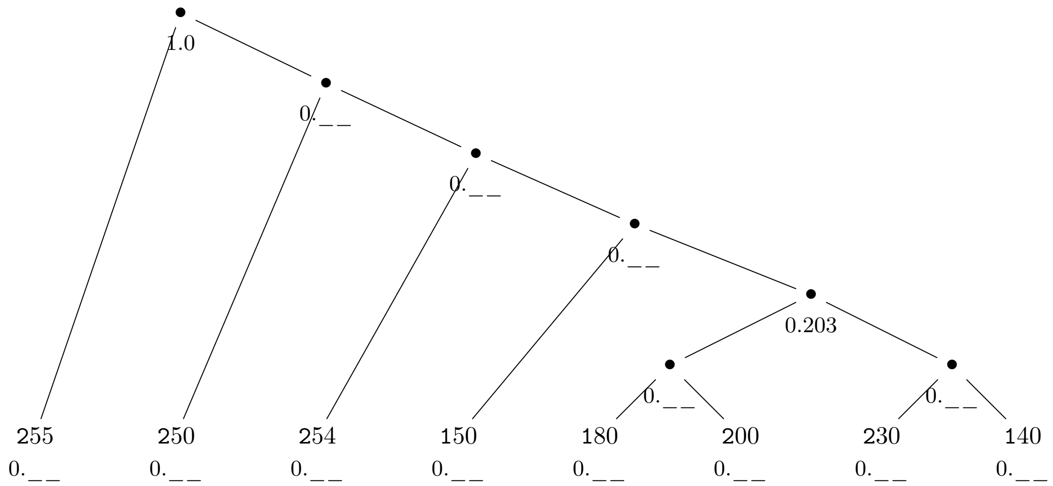
Step 7

Note: As no bit string is a prefix of any other bit string, this code is uniquely decodable.

Question 12 What is the word coded by the following code: `1110110010` ?

.... A E *For examiner only (do not check)*

In the case of the image Figure 2, the Huffman tree would look like presented in the Figure below.
Question 13 Complete the following Huffman tree with corresponding probabilities and branches encoding (0 for left branches and 1 for right branches).



.... A E *For examiner only (do not check)*

Question 14 Complete the following Huffman encoding table containing for each gray level its binary code and the length of this binary code.

gray level	140	150	180	200	230	250	254	255
binary code								
length l_i								

.... A B C D E *For examiner only (do not check)*

Question 15 What would be the final size (in *bits*) of the whole image of Figure 2 ?

.... A E *For examiner only (do not check)*

Question 16 What do you conclude ?

A B C D E *For examiner only (do not check)*

....

....

....

....

....

2.3 MP3 and JPEG encoding

To gain even more memory space when handling sound (1D) or image (2D) signals, one can also use lossy compression. For example, MP3 and JPEG format, before compressing using Huffman encoding suppress high frequencies on signals.

Question 17 How can one perform such an operation (give 2 methods) ?

☐A ☐B ☐C ☐D ☐E *For examiner only (do not check)*

Question 18 Why removing high frequencies in a signal would allow to sub-sample it ?

☐A ☐B ☐C ☐D ☐E *For examiner only (do not check)*