UE Image L3

TD-TP2 • Seuillage, Histogramme

TD Groupe 2 • 09.02.2021

Questions de cours

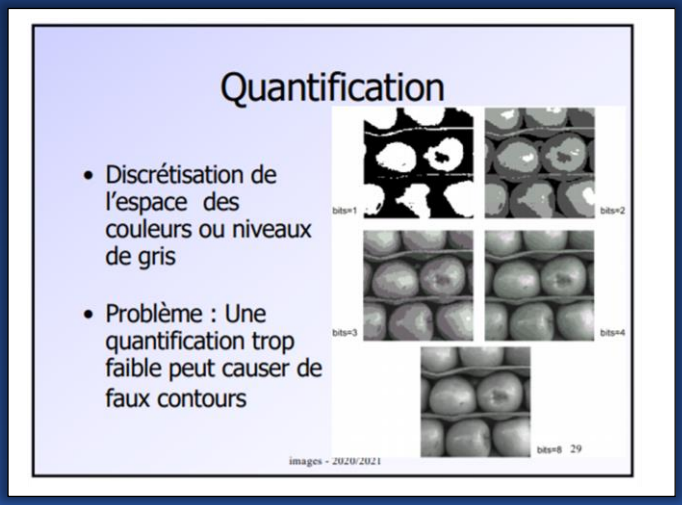
Quelles opérations peut-on faire sur un histogramme ?

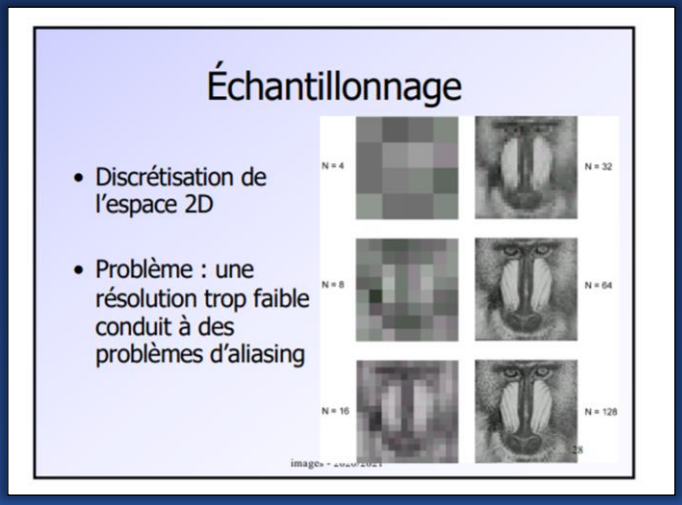
1. Un partage
2. Un seuillage
3. Une égalisation
4. Une quantification

Qu'est-ce que la quantification d'une image ?

1. Une discrétisation de l'espace 2D de l'image

2. Une discrétisation de l'espace de couleurs





Quels critères sont visés par le seuillage d'Otsu ?

1. Minimise la variance intra-classe

2. Maximise la variance intra-classe

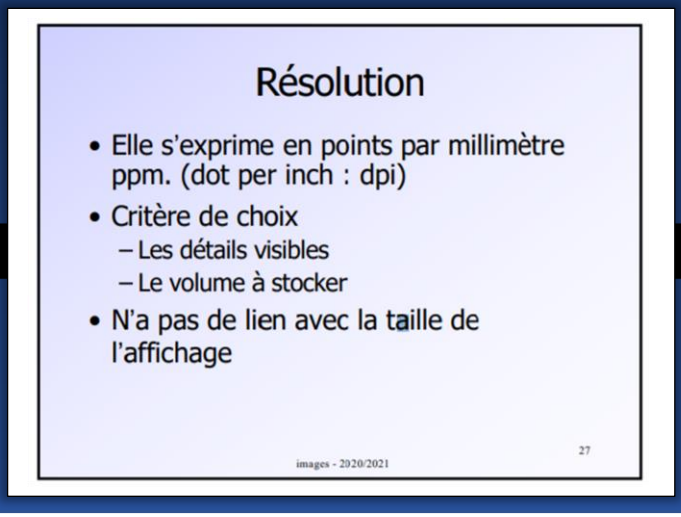
3. Minimise la variance inter-classe

4. Maximise la variance inter-classe

Dans la phrase « une image de 1920 par 1080 pixels », l'information « 1920 par 1080 pixels concerne :

1. La résolution de l'image
2. La taille de l'image

La résolution : nombre de pixels par unités de surface.



Licence informatique – 3ème année
UFR mathématique et informatique
Université de Paris

Université de Paris

TD-TP 2 : Analyse et traitement d'image

Exercice 1

1. Charger l'image « test.png ». Appliquer un seuillage sur la couleur en utilisant la valeur 145.
2. Que remarquez-vous ? Calculer l'aire de l'objet noir.
3. Faire varier le seuil et afficher l'évolution de l'aire en fonction.

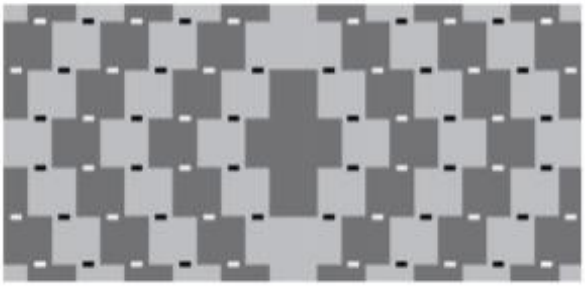
Exercice 2

1. Charger l'image « landscape.png ».
2. Calculer l'histogramme de cette image.
3. Binariser l'image en faisant varier le seuil.
4. Trouver un seuil qui n'extrait que les arbres et coloriez-les en vert.
5. Isolez au mieux les montagnes et coloriez-les en marron.

Exercice 3

Dans cet exercice, le but est de recréer l'illusion d'optique ci-dessous en suivant les étapes suivantes :

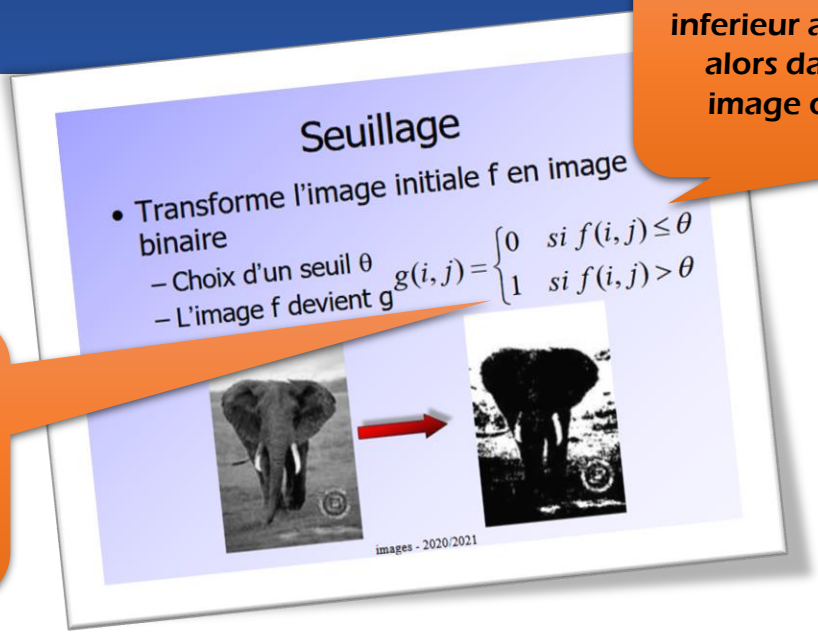
1. Créer une image synthétique de taille 630x1345pixels.
2. Dessiner sur cette image des carrés de taille 112x112 et les répéter avec 2 niveaux de gris (92 et 192) différents.
3. Répéter (2) afin de remplir l'image selon la disposition indiquée sur la figure ci-dessous.
4. Rajouter entre les carrés des petits traits de taille 8*20 pixels qui seront blanc ou noir selon la disposition indiquée sur la figure.



Exercice 1

(1) Charger l'image « test.png ». Appliquer un seuillage sur la couleur en utilisant le niveau à 145.

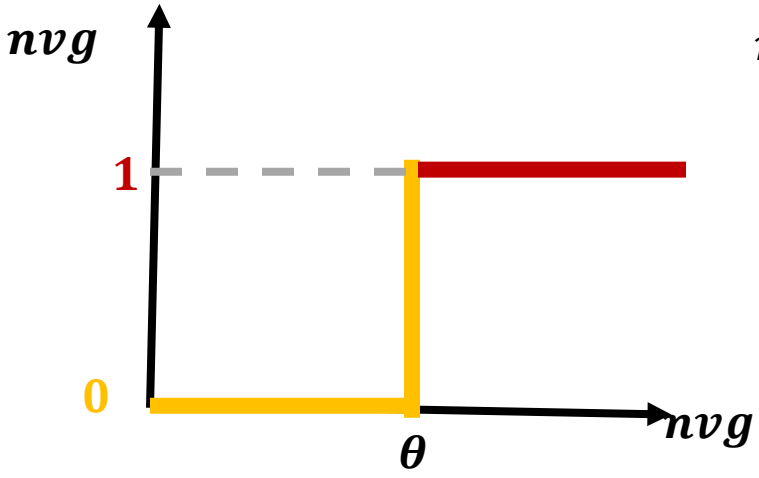
Si le niveau de gris est inferieur a la valeur Thêta, alors dans la nouvelle image on va mettre 0



Si le niveau de gris est supérieur à la valeur Thêta (donc est clair), alors dans la nouvelle image on va mettre 1

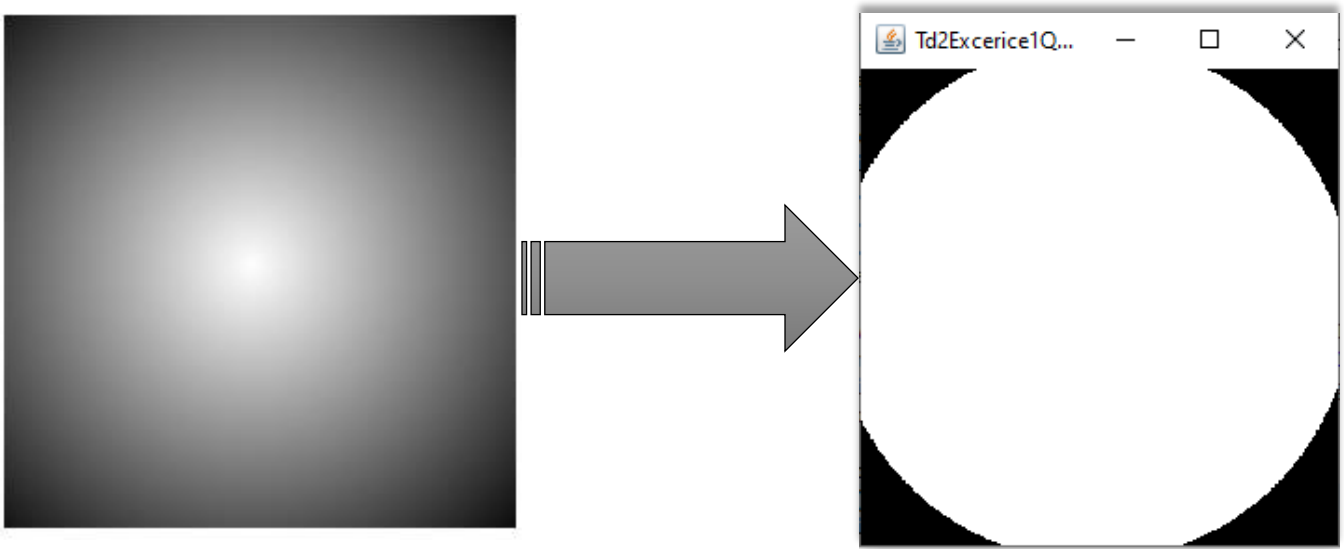
Cela nous permet de faire en sorte que tous ce qui est clair va avoir une certaine valeur, tous ce qui est plus foncer aura une autre valeur.

Donc, sur l'espace du niveau de gris je choisi Thêta, et en fonction de si je suis avant ou si je suis après, j'associe un niveau de gris qui est soit 0, soit 1.



$$nvg = Niveau\ de\ gris$$

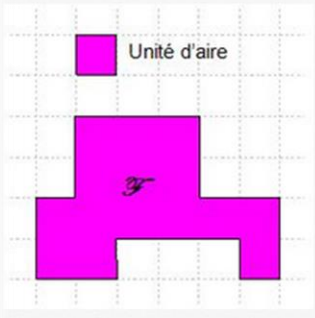
```java
import java.io.File;
import java.io.IOException;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.ImageIcon;
import javax.imageio.ImageIO;
import java.awt.Color;
import java.awt.image.BufferedImage;
public class Td2Exercice1Question1 {
	public static void showImage(BufferedImage bufferedImage) throws IOException {
		JFrame frame = new JFrame("Td2Exercice1Question1");
		ImageIcon imageIcon = new ImageIcon(bufferedImage);
		JLabel jLabel = new JLabel(imageIcon);

		frame.getContentPane().add(jLabel); // Set Content to the JFrame
		frame.pack();
		frame.setVisible(true);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	public static BufferedImage loadImage(File path) {
		BufferedImage img = null;

		try {
			img = ImageIO.read(path);
		} catch (IOException e) {
			e.printStackTrace();
		}

		return img;
	}

	public static void seuillage(BufferedImage img, int niveauSeuillage) {
		int nombre_col = img.getWidth();
		int nombre_lignes = img.getHeight();

		int noir = new Color(0, 0, 0).getRGB();
		int blanc = new Color(255, 255, 255).getRGB();
		int seuil_color = new Color(niveauSeuillage, niveauSeuillage, niveauSeuillage).getRGB();

		for(int x = 0 ; x < nombre_col ; x++) {
			for(int y = 0 ; y < nombre_lignes ; y++) {
				if(img.getRGB(x, y) < seuil_color) {
					img.setRGB(x, y, noir);
				}
				else {
					img.setRGB(x, y, blanc);
				} // else
			} // for(y)
		} // for(x)
	} // seuillage()

	public static void main(String[] args) {
		File path = new File("Test_Images" + File.separator + "test.png");
		BufferedImage img = loadImage(path);
		seuillage(img, 145);

		try {
			showImage(img);
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}
```



Exercice 1

(2) Que remarquez-vous ? Calculer l'aire de l'objet noir.

Définition de l'aire d'une surface



Définition :

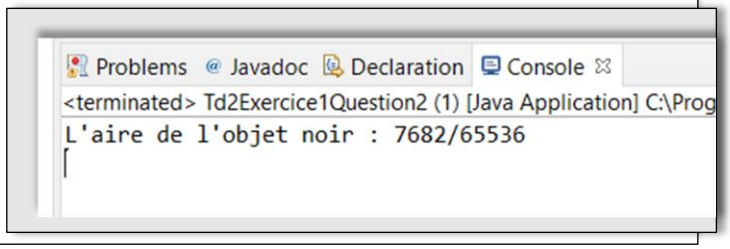
L'aire d'une surface est la mesure de la portion de plan recouverte par cette surface en fonction d'une unité d'aire choisie.

Exemple :

L'aire de la surface F est égale au nombre de carreaux recouverts par cette surface, soit : 15 unités d'aire.

Source : https://www.educastream.com/aire-surface-plane-5eme

```java
import java.io.File;
import java.io.IOException;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.ImageIcon;
import javax.imageio.ImageIO;
import java.awt.Color;
import java.awt.image.BufferedImage;
public class Td2Exercice1Question2 {
	public static void showImage(BufferedImage bufferedImage) throws IOException {
		JFrame frame = new JFrame("Td2Exercice1Question2");
		ImageIcon imageIcon = new ImageIcon(bufferedImage);
		JLabel jLabel = new JLabel(imageIcon);

		frame.getContentPane().add(jLabel); // Set Content to the JFrame
		frame.pack();
		frame.setVisible(true);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	public static BufferedImage loadImage(File path) {
		BufferedImage img = null;

		try {
			img = ImageIO.read(path);
		} catch (IOException e) {
			e.printStackTrace();
		}

		return img;
	}

	public static void seuillage(BufferedImage img, int niveauSeuillage) {
		int nombre_col = img.getWidth();
		int nombre_lignes = img.getHeight();

		int noir = new Color(0, 0, 0).getRGB();
		int blanc = new Color(255, 255, 255).getRGB();
		int seuil_color = new Color(niveauSeuillage, niveauSeuillage, niveauSeuillage).getRGB();
		int nb_pixels_noir = 0;
		for(int x = 0 ; x < nombre_col ; x++) {
			for(int y = 0 ; y < nombre_lignes ; y++) {
				if(img.getRGB(x, y) < seuil_color) {
					img.setRGB(x, y, noir);
					nb_pixels_noir++;
				}
				else {
					img.setRGB(x, y, blanc);
				} // else
			} // for(y)
		} // for(x)
		System.out.println("L'aire de l'objet noir : " + nb_pixels_noir + "/" + (nombre_col * nombre_lignes) );
	} // seuillage()

	public static void main(String[] args) {
		File path = new File("Test_Images" + File.separator + "test.png");
		BufferedImage img = loadImage(path);
		seuillage(img, 145);

		try {
			showImage(img);
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}
```



Exercice 1
(3) Faire varier le seuil et afficher l'évolution de l'aire en fonction.

```java
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import java.awt.Color;
import java.awt.image.BufferedImage;
public class Td2Exercice1Question3 {
	public static BufferedImage loadImage(File path) {
		BufferedImage img = null;

		try {
			img = ImageIO.read(path);
		} catch (IOException e) {
			e.printStackTrace();
		}

		return img;
	}

	public static void seuillage(BufferedImage img) {
		int nombre_col = img.getWidth();
		int nombre_lignes = img.getHeight();

		int noir = new Color(0, 0, 0).getRGB();
		int blanc = new Color(255, 255, 255).getRGB();

		BufferedImage new_img = new BufferedImage(nombre_col, nombre_lignes, BufferedImage.TYPE_3BYTE_BGR);
		for(int seuil = 0 ; seuil < 255 ; seuil++) {

			int nb_pixels_noir = 0;
			int seuil_color = new Color(seuil, seuil, seuil).getRGB();
			for(int x = 0 ; x < nombre_col ; x++) {
				for(int y = 0 ; y < nombre_lignes ; y++) {
					if(img.getRGB(x, y) < seuil_color) {
						new_img.setRGB(x, y, noir);
						nb_pixels_noir++;
					}
					else {
						new_img.setRGB(x, y, blanc);
					} // else
				} // for(y)
			} // for(x)
			System.out.println("Aire du noir pour le seuil " + seuil + " : " + nb_pixels_noir);
		} // for(seuil)
	} // seuillage()

	public static void main(String[] args) {
		File path = new File("Test_Images" + File.separator + "test.png");
		BufferedImage img = loadImage(path);
		seuillage(img);
	}
}
```

```
Seuil 0 : 0
Seuil 1 : 0
Seuil 2 : 0
Seuil 3 : 0
Seuil 4 : 0
Seuil 5 : 0
Seuil 6 : 0
Seuil 7 : 0
Seuil 8 : 0
Seuil 9 : 0
Seuil 10 : 0
Seuil 11 : 0
Seuil 12 : 0
Seuil 13 : 0
Seuil 14 : 0
Seuil 15 : 0
Seuil 16 : 0
Seuil 17 : 0
Seuil 18 : 0
Seuil 19 : 0
Seuil 20 : 0
Seuil 21 : 0
Seuil 22 : 0
Seuil 23 : 0
Seuil 24 : 0
Seuil 25 : 0
Seuil 26 : 0
Seuil 27 : 0
Seuil 28 : 0
Seuil 29 : 0
Seuil 30 : 0
Seuil 31 : 0
Seuil 32 : 0
Seuil 33 : 0
Seuil 34 : 0
Seuil 35 : 0
Seuil 36 : 0
Seuil 37 : 0
Seuil 38 : 0
Seuil 39 : 0
Seuil 40 : 0
Seuil 41 : 0
Seuil 42 : 0
Seuil 43 : 0
Seuil 44 : 0
Seuil 45 : 0
Seuil 46 : 0
Seuil 47 : 0
Seuil 48 : 0
Seuil 49 : 0
Seuil 50 : 0
Seuil 51 : 0
Seuil 52 : 0
```

```
Seuil 53 : 0
Seuil 54 : 0
Seuil 55 : 0
Seuil 56 : 0
Seuil 57 : 0
Seuil 58 : 0
Seuil 59 : 0
Seuil 60 : 0
Seuil 61 : 0
Seuil 62 : 1
Seuil 63 : 1
Seuil 64 : 1
Seuil 65 : 2
Seuil 66 : 2
Seuil 67 : 3
Seuil 68 : 3
Seuil 69 : 3
Seuil 70 : 9
Seuil 71 : 9
Seuil 72 : 15
Seuil 73 : 15
Seuil 74 : 21
Seuil 75 : 21
Seuil 76 : 29
Seuil 77 : 29
Seuil 78 : 39
Seuil 79 : 39
Seuil 80 : 61
Seuil 81 : 61
Seuil 82 : 73
Seuil 83 : 73
Seuil 84 : 94
Seuil 85 : 94
Seuil 86 : 120
Seuil 87 : 151
Seuil 88 : 151
Seuil 89 : 185
Seuil 90 : 185
Seuil 91 : 222
Seuil 92 : 222
Seuil 93 : 268
Seuil 94 : 316
Seuil 95 : 316
Seuil 96 : 365
Seuil 97 : 427
Seuil 98 : 427
Seuil 99 : 487
Seuil 100 : 555
Seuil 101 : 555
Seuil 102 : 626
Seuil 103 : 698
Seuil 104 : 698
Seuil 105 : 786
```

```
Seuil 106 : 872
Seuil 107 : 975
Seuil 108 : 975
Seuil 109 : 1075
Seuil 110 : 1178
Seuil 111 : 1291
Seuil 112 : 1291
Seuil 113 : 1396
Seuil 114 : 1517
Seuil 115 : 1652
Seuil 116 : 1770
Seuil 117 : 1770
Seuil 118 : 1912
Seuil 119 : 2057
Seuil 120 : 2206
Seuil 121 : 2355
Seuil 122 : 2528
Seuil 123 : 2693
Seuil 124 : 2693
Seuil 125 : 2859
Seuil 126 : 3027
Seuil 127 : 3220
Seuil 128 : 3408
Seuil 129 : 3596
Seuil 130 : 3813
Seuil 131 : 4004
Seuil 132 : 4229
Seuil 133 : 4446
Seuil 134 : 4693
Seuil 135 : 4929
Seuil 136 : 5156
Seuil 137 : 5421
Seuil 138 : 5666
Seuil 139 : 5925
Seuil 140 : 6199
Seuil 141 : 6476
Seuil 142 : 6757
Seuil 143 : 7058
Seuil 144 : 7382
Seuil 145 : 7682
Seuil 146 : 8008
Seuil 147 : 8341
Seuil 148 : 8678
Seuil 149 : 9408
Seuil 150 : 9780
Seuil 151 : 10166
Seuil 152 : 10568
Seuil 153 : 10975
Seuil 154 : 11430
Seuil 155 : 11895
Seuil 156 : 12875
Seuil 157 : 13410
Seuil 158 : 13915
```

```
Seuil 159 : 14419
Seuil 160 : 15507
Seuil 161 : 16029
Seuil 162 : 16603
Seuil 163 : 17193
Seuil 164 : 18381
Seuil 165 : 18960
Seuil 166 : 19539
Seuil 167 : 20132
Seuil 168 : 21286
Seuil 169 : 21841
Seuil 170 : 22431
Seuil 171 : 23533
Seuil 172 : 24082
Seuil 173 : 24643
Seuil 174 : 25724
Seuil 175 : 26268
Seuil 176 : 27343
Seuil 177 : 27877
Seuil 178 : 28408
Seuil 179 : 29456
Seuil 180 : 29956
Seuil 181 : 30966
Seuil 182 : 31470
Seuil 183 : 32456
Seuil 184 : 32942
Seuil 185 : 33444
Seuil 186 : 34415
Seuil 187 : 34879
Seuil 188 : 35836
Seuil 189 : 36301
Seuil 190 : 37213
Seuil 191 : 38128
Seuil 192 : 38559
Seuil 193 : 39452
Seuil 194 : 39891
Seuil 195 : 40748
Seuil 196 : 41184
Seuil 197 : 42025
Seuil 198 : 42836
Seuil 199 : 43266
Seuil 200 : 44028
Seuil 201 : 44830
Seuil 202 : 45230
Seuil 203 : 46004
Seuil 204 : 46752
Seuil 205 : 47122
Seuil 206 : 47834
Seuil 207 : 48557
Seuil 208 : 48889
Seuil 209 : 49596
Seuil 210 : 50296
Seuil 211 : 50942
```

```
Seuil 159 : 14419
Seuil 160 : 15507
Seuil 161 : 16029
Seuil 212 : 51280
Seuil 213 : 51901
Seuil 214 : 52549
Seuil 215 : 53148
Seuil 216 : 53759
Seuil 217 : 54329
Seuil 218 : 54600
Seuil 219 : 55189
Seuil 220 : 55717
Seuil 221 : 56254
Seuil 222 : 56771
Seuil 223 : 57268
Seuil 224 : 57745
Seuil 225 : 58222
Seuil 226 : 58462
Seuil 227 : 58895
Seuil 228 : 59347
Seuil 229 : 59756
Seuil 230 : 60173
Seuil 231 : 60559
Seuil 232 : 60927
Seuil 233 : 61288
Seuil 234 : 61631
Seuil 235 : 61968
Seuil 236 : 62289
Seuil 237 : 62585
Seuil 238 : 62882
Seuil 239 : 63281
Seuil 240 : 63528
Seuil 241 : 63761
Seuil 242 : 63989
Seuil 243 : 64186
Seuil 244 : 64382
Seuil 245 : 64565
Seuil 246 : 64725
Seuil 247 : 64940
Seuil 248 : 65071
Seuil 249 : 65174
Seuil 250 : 65271
Seuil 251 : 65354
Seuil 252 : 65447
Seuil 253 : 65485
Seuil 254 : 65518
```

Exercice 2

Vocabulaire correcte

L'histogramme de l'image

~~L'histogramme des niveau du gris dans l'image~~

Combien y'a de pixels qui sont au niveau 0,

Combien y'a de pixel qui sont au niveau 1

Combien de pixel sont au niveau n-1

n - 1 car il y a n niveaux et on commence par 0..

On va regrouper tt les pixels qui ont un niveau entre 0 et 10, entre 10 et 20 et ainsi suite.. (= définir des classes)

On va analyser l'histogramme pour essayer de trouver Thêta (niveau seuillage)

Au lieu de dire : nombre de pixel 0, 1, ... n-1 , on va dire nombre de pixel inferieur a 25, nombre de pixel inferieur a 26, donc on a quelque chose qui va être une fonction croissante et comprise entre 0 et 1.

0 au départ parce que y'a rien qui est strictement inferieur a 0 , et pour strictement inferieur a n : on a tout le monde, donc on a 1.

On parle de la dynamique de l'histogramme en regardant quelle est la valeur minimum et quelle est la valeur maximum des classes qui ne sont pas vide.

On obtient de la saturation quand le capteur ne peut plus distinguer les couleurs.

Sur un histogramme ça se traduit par un pic à droite ou à gauche.

En traitement d'image quand on parle d'histogramme, on parle de nombre de pixels, mais le nombre de pixels dépond de la taille de l'image, donc on parle d'histogramme normalisé quand la somme des fréquences est égale à 1.

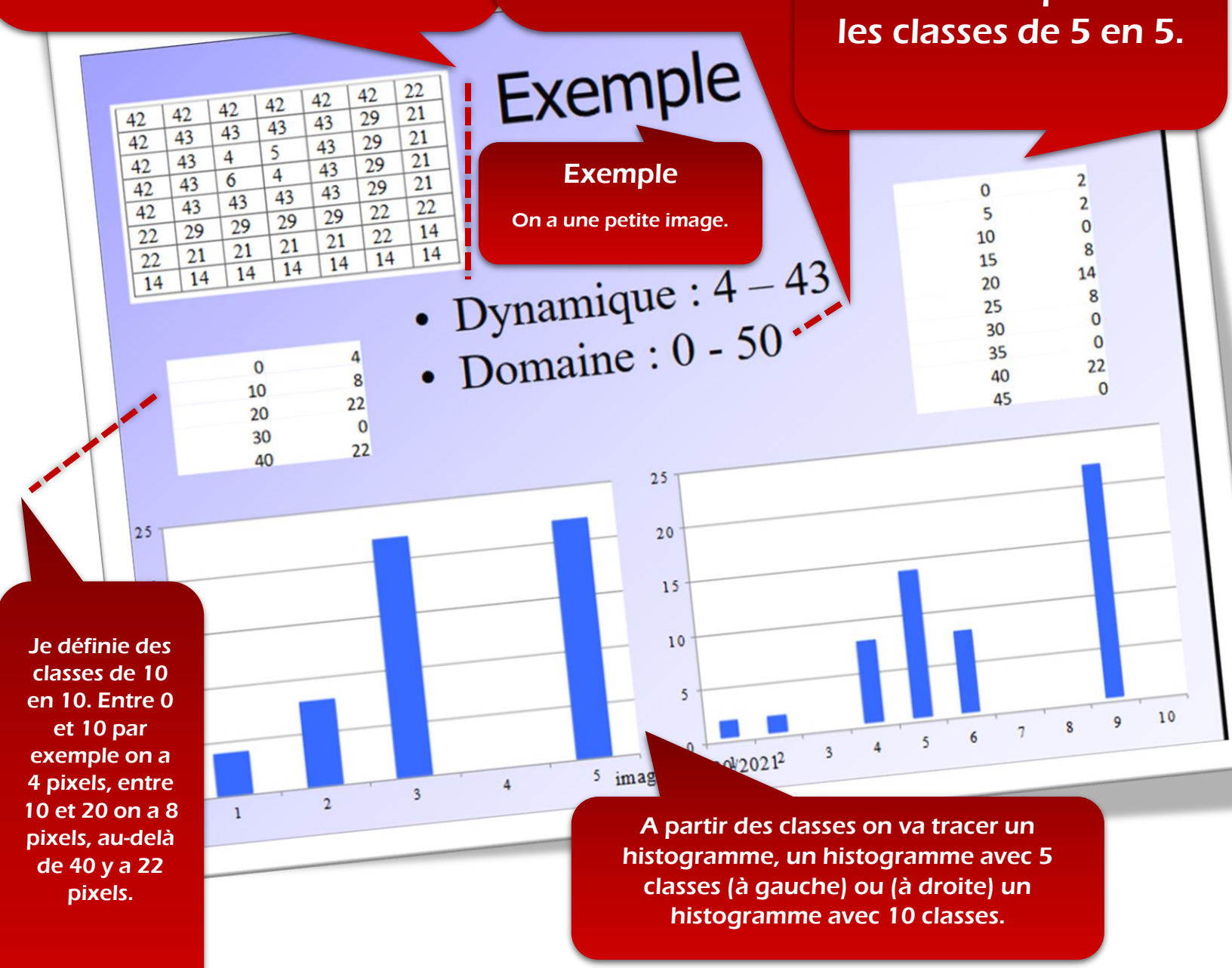
Donc on divise chaque cardinal de classe par le nombre total de pixels de l'image, qui fait que on a que des nombres qui sont compris entre 0 et 1 au maximum, (1 supposerai que tous les pixels sont de même couleur).

Dynamique

je regarde quelle est la valeur du pixel minimum, et quelle est la valeur du pixel maximum. Donc ici la dynamique est de 4-43. (L'intervalle de dynamique).

Ici on va dire que le domaine possible pour les valeurs c'est compris entre 0 et 50.

Une façon de définir les classes : prendre les classes de 5 en 5.



Je définie des classes de 10 en 10. Entre 0 et 10 par exemple on a 4 pixels, entre 10 et 20 on a 8 pixels, au-delà de 40 y a 22 pixels.

A partir des classes on va tracer un histogramme, un histogramme avec 5 classes (à gauche) ou (à droite) un histogramme avec 10 classes.

```java
import java.io.File;
import java.io.IOException;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;

import java.awt.Color;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;

public class Td2Exercice2 {
	public static void imgShow(BufferedImage image) throws IOException {
		// Initiate JFrame
		JFrame frame = new JFrame();

		// Set Content to the JFrame
		frame.getContentPane().add(new JLabel(new ImageIcon(image)));
		frame.pack();
		frame.setVisible(true);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}


	public static void exo2() {
		File path = new File("Test_Images" + File.separator + "landscape.png");

		BufferedImage img = null;

		try {
			img = ImageIO.read(path);
		} catch (IOException e) {
			e.printStackTrace();
		}

		int nombre_col = img.getWidth();
		int nombre_lignes = img.getHeight();

		int histo[] = new int[255];
		for(int i = 0 ; i < 255 ; i++)
			histo[i] = 0;

		for(int x = 0 ; x < nombre_col ; x++) {
			for(int y = 0 ; y < nombre_lignes ; y++) {
				int color = img.getRGB(x, y) & 0xff;
				histo[color]++;
			}
		}

		for(int i = 0 ; i < 255 ; i++)
			System.out.println(i + "i " + histo[i]);

		// Seuil
		BufferedImage new_img = new BufferedImage(nombre_col, nombre_lignes, BufferedImage.TYPE_3BYTE_BGR);

		int noir = new Color(0, 0, 0).getRGB();
		int blanc = new Color(255, 255, 255).getRGB();
		int seuil = 150;

		for(int x = 0 ; x < nombre_col ; x++) {
			for(int y = 0 ; y < nombre_lignes ; y++) {
				int color = img.getRGB(x, y) & 0xff;
				if(color < seuil)
					new_img.setRGB(x, y, noir);
				else
					new_img.setRGB(x, y, blanc);
			} // for(y)
		} // for(x)
```

```
                // Coloriage
                int vert = new Color(0, 255, 0).getRGB();
                seuil = 150;
                for(int x = 0 ; x < nombre_col ; x++) {
                        for(int y = 0 ; y < nombre_lignes ; y++) {
                                int color = img.getRGB(x, y);
                                int colorb = color & 0xff;
                                if( colorb < seuil)
                                        new_img.setRGB(x, y, vert);
                                else
                                        new_img.setRGB(x, y, color);
                        }
                }

                //int seuil1 = 162;
                //int seuil2 = 173;
                //int marron = new Color(150, 113, 23).getRGB();
                //for(int x = 0 ; x < nombre_col ; x++) {
                        //for(int y = 0 ; y < nombre_lignes ; y++) {
                        //      int color = img.getRGB(x, y);
                        //      int colorb = color & 0xff;
                        //      if(colorb < seuil2 && colorb > seuil1)
                        //              new_img.setRGB(x, y, marron);
                        //      else
                        //              new_img.setRGB(x, y, color);
                        //}
                //}

                // Affichage de l'image
                try {
                        imgShow(new_img);
                } catch(IOException e) {
                        e.printStackTrace();
                }
        } // exo2()

        public static void main(String[] args) {
                exo2();
        }
} // class
```

Exercice 3

```java
import java.io.IOException;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;

import java.awt.Color;
import java.awt.image.BufferedImage;
public class Td2Exercice3 {
	public static void imgShow(BufferedImage image) throws IOException {
		// Initiate JFrame
		JFrame frame = new JFrame();

		// Set Content to the JFrame
		frame.getContentPane().add(new JLabel(new ImageIcon(image)));
		frame.pack();
		frame.setVisible(true);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	public static void exo3() {
		int width = 1345;
		int height = 630;
		int square_size = 112;
		BufferedImage img = new BufferedImage(width, height, BufferedImage.TYPE_3BYTE_BGR);

		int g1 = new Color(92, 92, 92).getRGB();
		int g2 = new Color(192, 192, 192).getRGB();
		int noir = new Color(0, 0, 0).getRGB();
		int blanc = new Color(255, 255, 255).getRGB();

		int first_sq_height = (height % square_size) / 2;

		// Background
		int y = 0;
		int yend = 0;
		int nline = 0;
		while(y < height - 1) {
			int x = 0;
			int xend = 0;
			int color = g1;
			if(nline == 0 || nline == 3 || nline == 6)
				color = g2;
			int ncol = 0;

			while(x < width - 1) {
				if(nline % 2 == 0 && x == 0)
					xend = x + square_size / 2;
				else
					xend = x + square_size;
				if(xend > width - 1)
					xend = width - 1;

				if(y == 0)
					yend = y + first_sq_height;
				else
					yend = y + square_size;
				if(yend > height - 1)
					yend = height - 1;

				for(int x2 = x ; x2 < xend ; x2++) {
					for(int y2 = y ; y2 < yend ; y2++) {
						img.setRGB(x2, y2, color);
					}
				}
```

```
                        if(ncol != 5 || nline % 2 == 0) {
                                if(color == g2)
                                        color = g1;
                                else
                                        color = g2;
                        }
                        x = xend;
                        ncol++;
                }
                y = yend;
                nline++;
        }

        // Rectangles
        y = first_sq_height - 4;
        nline = 0;
        while(y < height - 1) {
                int x = 0;
                int ncol = 0;
                if(nline == 1 || nline == 4)
                        x = square_size / 4 - 10;
                else
                        x = square_size * 3 / 4 - 10;

                int color = blanc;
                if(nline == 2 || nline == 3)
                        color = noir;

                while(x < width - 1) {
                        for(int x2 = x ; x2 < x + 20 ; x2++) {
                                for(int y2 = y ; y2 < y + 8 ; y2++) {
                                        img.setRGB(x2, y2, color);
                                }
                        }
                        if (color == noir)
                                color = blanc;
                        else
                                color = noir;
                        x += square_size;
                        ncol++;
                        if((nline == 0 || nline == 5) && ncol == 5) {
                                x += square_size * 1.5;
                                color = blanc;
                        }
                        if((nline == 1 || nline == 4) && ncol == 6) {
                                x += square_size * .5;
                                color = noir;
                        }
                        if((nline == 2 || nline == 3) && ncol == 5) {
                                x += square_size * 1.5;
                                color = noir;
                        }
                }
                y += square_size;
                nline++;
        }

        // Affichage de l'image
        try {
                imgShow(img);
        } catch(IOException e) {
                e.printStackTrace();
        }
    } // exo3()

    public static void main(String[] args) {
            exo3();
    }
} // class
```