UE Image L3

TD-TP3

TD Groupe 2 • 16.02.2021

Questions de cours

Quelles affirmations sont exactes ?

1 . Un filtre passe bas supprime des détails.

2 . Un filtre passe bas réhausse des détails

3 . Un filtre passe haut supprime des détails

4 . Un filtre passe haut réhausse des détails

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.

Une opération ponctuelle est... ?

1. Une opération qu'on ne peut lancer qu'une fois par image.
2. Une opération au niveau du pixel.
3. Une opération pour mettre en valeur des points

- Ponctuelle, c'est le mot point, et le point d'une image c'est un pixel.
- De plus, une opération qu'on ne peut lancer qu'une fois par image – ça n'existe pas.
- Une opération pour mettre en valeur des points : ça existe, mais c'est l'inverse d'une méthode ponctuelle, ça va plutôt être une méthode qui va devoir regarder le voisinage du pixel.

La convolution est (généralement) une méthode ?

1. **Locale**
2. Globale
3. Ponctuelle

- Globale : regarder l'image dans son intégralité.
- La convolution est une opération locale car on regarde un voisinage défini sur le nombre de noyaux de convolution (noyaux de convolution c'est par exemple un masque 3 par 3) et donc c'est ce qu'on appelle une opération local parce qu'on regarde l'image localement.

Exercice 1

Soit l'image suivante en niveaux de gris (compris entre 1 et 100) :

12	10	8	13	25	50	46
8	7	8	10	20	23	31
11	9	10	14	28	30	37
14	11	10	26	31	28	32
12	9	11	31	35	33	41
7	12	33	35	41	50	70
13	10	35	38	75	73	72
9	14	41	45	71	76	75

(1) En considérant des classes d'amplitude (= variation) 10, tracer l'histogramme de cette image.

```java
import java.util.Map;
import java.util.Arrays;
import java.util.HashMap;
public class Td3Exercice1Question1 {

	// Amplitude = variation
	public static Map<Integer, Integer> histogram(int[][] niveauxDeGris, int amplitudeClasses) {
		Map<Integer, Integer> histro = new HashMap<Integer, Integer>();

		for(int i = 1 ; i < 100 ; i += amplitudeClasses)
			histro.put(i, 0); // put(key, value)

		for(int i = 0 ; i < niveauxDeGris.length ; i++) {
			for(int j = 0 ; j < niveauxDeGris[i].length ; j++) {
				int color = niveauxDeGris[i][j];
				int classe = 1;
				for(int c = 1 ; c < 100 ; c += amplitudeClasses) { // c = classe
					if(color >= c) classe = c;
				}

				int newValue = (histro.get(classe)) + 1;
				histro.replace(classe, newValue);
			}
		}
		return histro;
	}

	public static void main(String[] args) {
		int[][] niveauxDeGris = { {12, 10, 8, 13, 25, 50, 46},
		                          {8, 7, 8, 10, 20, 23, 31},
		                          {11, 9, 10, 14, 28, 30, 37},
		                          {14, 11, 10, 26, 31, 28, 32},
		                          {12, 9, 11, 31, 35, 33, 41},
		                          {7, 12, 33, 35, 41, 50, 70},
		                          {13, 10, 35, 38, 75, 73, 72},
		                          {9, 14, 41, 45, 71, 76, 75} };

		Map<Integer, Integer> histro = histogram(niveauxDeGris, 10);
		Object[] classes =  histro.keySet().toArray();
		Arrays.sort( classes );

		for(Object classe : classes)
			System.out.println( classe + "-" + (Integer.parseInt(classe.toString()) + 9) + " : " +
histro.get(classe));
	}
}
```

Problems | Javadoc | Declaration | Console

<terminated> Td3Exercice1Question1 [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (Feb 28, 2021, 9:55:56 PM –

```
1-10 : 13
11-20 : 12
21-30 : 6
31-40 : 11
41-50 : 7
51-60 : 0
61-70 : 1
71-80 : 6
81-90 : 0
91-100 : 0
```

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.

Exemple

42	42	42	42	42	42	22
42	43	43	43	43	29	21
42	43	4	5	43	29	21
42	43	6	4	43	29	21
42	43	43	43	43	29	21
22	29	29	29	29	22	22
22	21	21	21	21	22	14
14	14	14	14	14	14	14

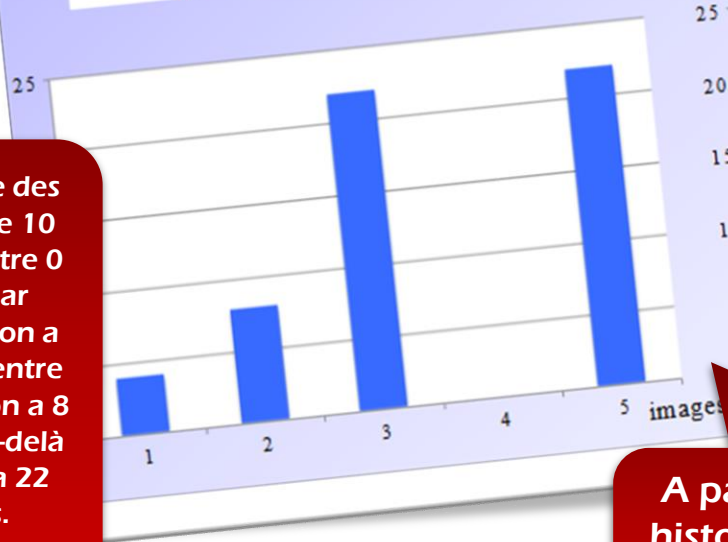
Exemple

On a une petite image.

- Dynamique : 4 – 43
- Domaine : 0 - 50

0	4
10	8
20	22
30	0
40	22

0	2
5	2
10	0
15	8
20	14
25	8
30	0
35	0
40	22
45	0



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.

Exercice 1

(2) Que pensez-vous de la qualité de la prise d'image ? Quelles modifications proposeriez-vous d'apporter pour améliorer l 'image ? Le réaliser.

Egalisation histogramme.

Exercice 2

```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 Td3Exercice2 {
	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[256];
		int histro_cum[] = new int[256];
		for(int i = 0 ; i < 256 ; 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]++;
			}
		}

		histro_cum[0] = histo[0];
		for(int i = 1 ; i < 256 ; i++)
			histro_cum[i] = histro_cum[i-1] + histo[i];

		for(int i = 0 ; i < 256 ; i++)
			System.out.println(i + "i " + histo[i] + " cum: " + histro_cum[i]);

		int transfo[] = new int[256];
		int n = 256;
		int N = nombre_col * nombre_lignes;
		for(int i = 0 ; i < 256 ; i++)
			transfo[i] = Math.max(0, n * histro_cum[i] / N - 1);
```

```
BufferedImage eq_img = new BufferedImage(nombre_col, nombre_lignes, BufferedImage.TYPE_3BYTE_BGR);
for(int x = 0 ; x < nombre_col ; x++) {
        for(int y = 0 ; y < nombre_lignes ; y++) {
                int original_level = img.getRGB(x, y) & 0xff;
                int new_level = transfo[original_level];
                int new_color = new Color(new_level, new_level, new_level).getRGB();
                eq_img.setRGB(x, y, new_color);
        }
}


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

// 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();
                }

                //Otsu
                int meilleur_seuil = 0;
                double minimum = 1000000000000000.0;
                for(int seuilo = 0 ; seuilo < 255 ; seuilo++) {
                        // Calcul poids et moyenne
                        float w1 = 0;
                        float w2 = 0;
                        float mu1 = 0;
                        float mu2 = 0;
                        for(int i = 0 ; i < seuilo ; i++)
                        {
                                w1 += histo[i];
                                mu1 += i * histo[i];
                        }
                        for(int i = seuilo ; i < 255 ; i++)
                        {
                                w2 += histo[i];
                                mu2 += i * histo[i];
                        }
                        if(w1 == 0 || w2 == 0)
                                continue;
                        mu1 /= w1;
                        mu2 /= w2;
                        w1 /= 1.0 * nombre_col * nombre_lignes;
                        w2 /= 1.0 * nombre_col * nombre_lignes;

                        // Variance
                        float s1 = 0;
                        float s2 = 0;
                        for(int i = 0 ; i < seuilo ; i++)
                                s1 += (i - mu1) * (i - mu1) * histo[i];
                        for(int i = seuilo ; i < 255 ; i++)
                                s2 += (i - mu2) * (i - mu2) * histo[i];

                        float intra_class_var = w1 * s1 + w2 * s2;
                        // System.out.println("w1 = " + w1 + "w2 = " + w2 + "s1 = " + s1 + " s2 = " + s2);
                        if(intra_class_var < minimum) {
                                minimum = intra_class_var;
                                meilleur_seuil = seuilo;
                                System.out.println("Nouveau meilleur seuil à " + seuilo + ", score ");
                        }
                }
                System.out.println("Seuil : " + meilleur_seuil);
                try {
                        imgShow(new_img);
                } catch(IOException e) {
                        e.printStackTrace();
                }
        } // exo2()

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

Exercice 3

(2) Transformer cette image couleur en niveaux 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 Td3Exercice3Question2 {
	public static void showImage(BufferedImage bufferedImage) throws IOException {
		JFrame frame = new JFrame("Td3Exercice3Question2");
		ImageIcon imageIcon = new ImageIcon(bufferedImage);
		JLabel jLabel = new JLabel(imageIcon);

		frame.getContentPane().add(jLabel);
		frame.pack();
		frame.setVisible(true);
	}

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

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

		return img;
	}

	public static BufferedImage transformerImageCouleurEnNiveauxDeGris(BufferedImage imageCouleur) {
		int nombre_col = imageCouleur.getWidth();
		int nombre_lignes = imageCouleur.getHeight();

		BufferedImage g_img = new BufferedImage(nombre_col, nombre_lignes, BufferedImage.TYPE_3BYTE_BGR);

		/* Parcourir tous les pixels,
		 * pour chaque pixel extraire les valeurs r, g, b.
		 * La valeur du niveau de gris : la moyenne (r,g,b)/3
		 */

		for(int x = 0 ; x < nombre_col ; x++) {
			for(int y = 0 ; y < nombre_lignes ; y++) {
				int b = imageCouleur.getRGB(x, y) & 0xff;
				int g = (imageCouleur.getRGB(x,y) >> 8) & 0xff;
				int r = (imageCouleur.getRGB(x,y) >> 16) & 0xff;

				int level = (b + g + r) / 3;
				int level_c = new Color(level, level, level).getRGB();
				g_img.setRGB(x, y, level_c);
			}
		}
		return g_img;
	}

	public static void main(String[] args) {
		File path = new File("Test_Images" + File.separator + "shapes.jpg");
		BufferedImage imageCouleur = loadImage(path);
		BufferedImage imageEnNiveauxDeGris = transformerImageCouleurEnNiveauxDeGris(imageCouleur);

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

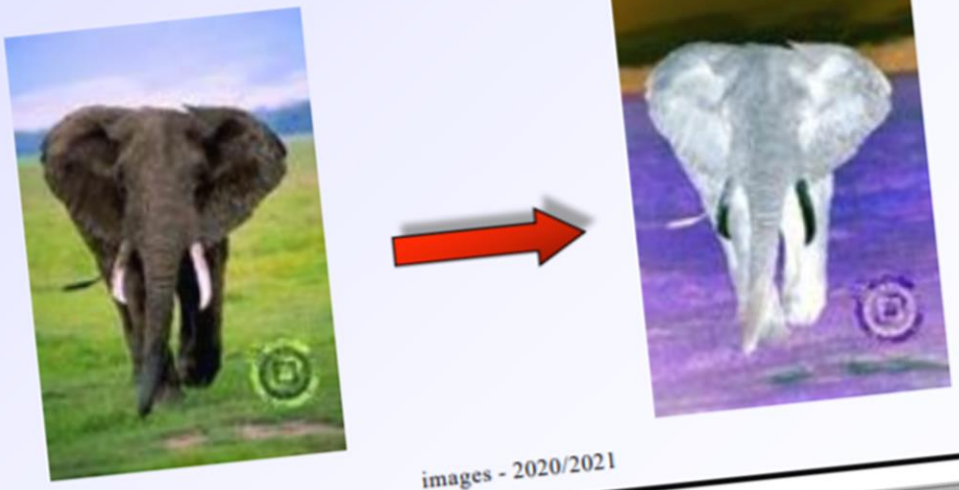
Exercice 3

(3) Générer l' image négative de l' image couleur.

Image négative

- Sur une image couleur

s(r,g,b) devient de couleur (255-r, 255-g, 255-b)



```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 Td3Exercice3Question3 {
	public static void showImage(BufferedImage bufferedImage) throws IOException {
		JFrame frame = new JFrame("Td3Exercice3Question3");
		ImageIcon imageIcon = new ImageIcon(bufferedImage);
		JLabel jLabel = new JLabel(imageIcon);

		frame.getContentPane().add(jLabel);
		frame.pack();
		frame.setVisible(true);
	}

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

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

		return img;
	}

	public static BufferedImage genererImageNegativeDeImageCouleur(BufferedImage imageCouleur) {
		int nombre_col = imageCouleur.getWidth();
		int nombre_lignes = imageCouleur.getHeight();

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

		for(int x = 0 ; x < nombre_col ; x++) {
			for(int y = 0 ; y < nombre_lignes ; y++) {
				int b = imageCouleur.getRGB(x, y) & 0xff;
				int g = (imageCouleur.getRGB(x,y) >> 8) & 0xff;
				int r = (imageCouleur.getRGB(x,y) >> 16) & 0xff;

				int new_color = new Color(255 - r, 255 - g, 255 - b).getRGB();
				new_img.setRGB(x, y, new_color);
			}
		}
		return new_img;
	}

	public static void main(String[] args) {
		File path = new File("Test_Images" + File.separator + "shapes.jpg");
		BufferedImage imageCouleur = loadImage(path);
		BufferedImage imageEnNiveauxDeGris = genererImageNegativeDeImageCouleur(imageCouleur);

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