UE Image L3

TD-TP1

TD Groupe 2 • 02.02.2021

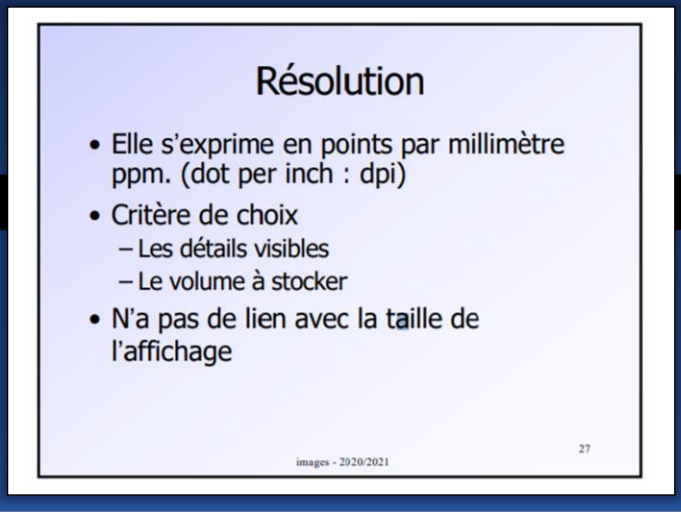
Questions de cours

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

1 . La résolution de l'image

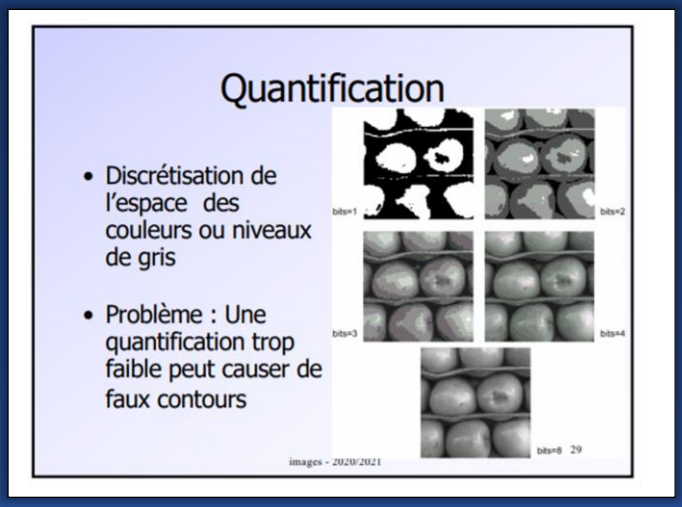
2 . La taille de l'image

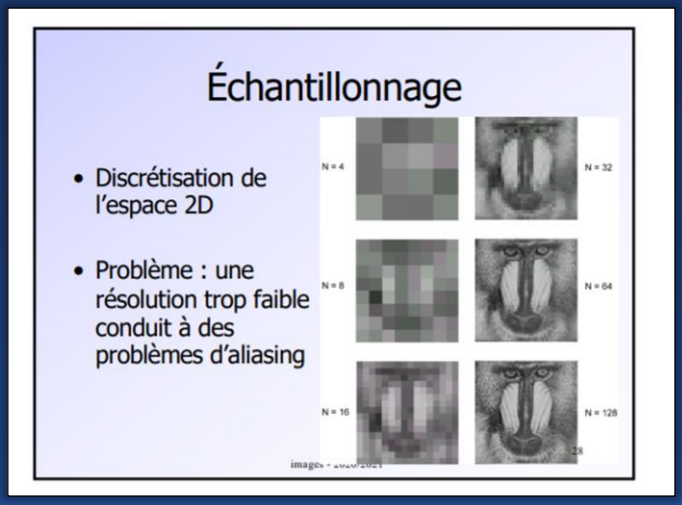
La résolution : nombre de pixels par unités de surface.



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**





Quel(s) terme(s) désignent des espaces de couleur ?

1. **RGB**
2. ESS
3. **HSV**
4. RSB

Teinte Saturation Valeur

Pour les articles homonymes, voir TSV, HSV et HSB.

Le modèle **TSV** pour **Teinte Saturation Valeur** (en anglais **HSV** pour *Hue Saturation Value* ou **HSB** pour *Hue Saturation Brightness*), est un système de gestion des couleurs en informatique.

Il fait partie d'une famille de systèmes basés sur la perception des couleurs, basés sur trois composantes définies par une approche psychologique et perceptuelle de la couleur : teinte, saturation et valeur.

La binarisation d'une image est un traitement...

1. **bas niveau**
2. haut niveau

Bas niveau = Au niveau du pixel binarisation via Seuillage -> bas niveau

Que fait-on avec une image ?

- Analyse d' images
- Haut niveau
  - Compréhension de scène
  - Réduction d' information
  - Décision d'action
- Bas niveau
  - Comparaison
  - Extraction de contours
  - Extraction de paramètres

Exercice 0 (prise en main)

(1) Charger une des images dans le dossier Test_Images et afficher là.

Vous utiliserez la classe **java.awt.image.BufferedImage**

Manipuler des images en Java

- Java AWT (Abstract Window Toolkit) -> API pour la création d'interface graphique en Java
- Classe java.awt.image pour représenter une image dans une GUI:
  - Contient informations telles que: type, taille, ... et les données (raster)
  - Java.awt.image -> données non accessibles
- Classe java.awt.BufferedImage -> les données sont temponnées (et donc accessibles)
- Constructeur à utiliser: BufferedImage(int width, int height, int imageType)
- imageType peut être:
  - TYPE_3BYTE_BGR
  - TYPE_4BYTE_ABGR
  - TYPE_BYTE_BINARY
  - TYPE_BYTE_GRAY
  - ...

- Taille: img.getWidth() et img.getHeight()
- Accèder aux données: img.getRaster() (objet de classe WritableRaster)
- Lire un pixel de l'image: **img.getRaster()**.getPixel(int x, int y)
- Ecrire un pixel de l'image **img.getRaster()**.setPixel(int x, int y, int[] couleur)
  - Couleur dépend du type de données. Pour ARGB: **int**[] bleu = {0,0,255,255};
- Dans le cas de RGB, accès direct:
  - Img.getRGB(int x, int y)
  - Img.setRGB(int x, int y, **int** couleur) -> Attention, couleur codée par octet sur un entier (voir exemple TD1)

- Classe javax.imageio.ImageIO -> swing aussi pour la creation d'interfaces utilisateur
- ImageIO.read(obj) -> obj peut-être:
  - File (pour un fichier en local)
  - URL (pour une image depuis internet)
- Possibilité de sauvegarder l'image avec ImageIO.write

Source : https://www.sylvainlobry.com/wp-content/uploads/2021/02/Semaine2.pdf

```java
import javax.imageio.ImageIO;
import java.io.*; // java.io.File, java.io.IOException
import javax.swing.*; // javax.swing.ImageIcon, javax.swing.JFrame, javax.swing.JLabel
import java.awt.image.BufferedImage;

public class Exo0a {
	public static void showImage(BufferedImage bufferedImage) throws IOException {
		// javax.swing.JFrame.JFrame(String title).
		JFrame frame = new JFrame("Exo0a"); //Instantiate JFrame , Constructs a new frame that is initially invisible.

		/* javax.swing.ImageIcon.ImageIcon(Image image)
		 * Creates an ImageIcon from an image object.If the image has a "comment" property that is a string,
		 * then the string is used as the description of this icon.
		 */
		ImageIcon imageIcone = new ImageIcon(bufferedImage);
		JLabel jLabel = new JLabel(imageIcone);

		//Set Content to the JFrame
		frame.getContentPane().add(jLabel); // Appends the specified component to the end of this container.

		/* Causes this Window to be sized to fit the preferred size
		 * and layouts of its subcomponents.
		 * The resulting width and height of the window are automatically enlarged if either of
		 * dimensions is less than the minimum size as specified by the previous call to the setMinimumSize method.
		 */
		frame.pack();

		// void java.awt.Window.setVisible(boolean b)
		// Shows or hides this Window depending on the value of parameter b.
		frame.setVisible(true);
	}

	public static void main(String[] args) {
		/* Manipulation de fichiers : la classe File
		 * File  permet de manipuler des fichiers et des repertoires grace au chemin qui les identifie.
		 * L'attribut separator permet de construire un chemin en utilisant les separateur adapte
		 * au systeme d'exploitation (/ sous Unix,\\ sous Windows).
		 * Source : Cours Programmation Avancee et Application / Jean-Guy Mailly.
		 */
		File path = new File("Test_Images" + File.separator + "text1.jpg");

		BufferedImage img = null; // Initialisation de l'objet BufferedImage.

		try {
			// BufferedImage javax.imageio.ImageIO.read(File input) throws IOException
			img = ImageIO.read(path);
		} catch (IOException e) {
			e.printStackTrace(); // Prints this throwable and its backtrace to the standard error stream.
		}

		int nombre_col  = img.getWidth(); // int.BufferedImage.getWidth() Returns the width of the BufferedImage.
		int nombre_lignes = img.getHeight(); // int BufferedImage.getHeight() Returns the height of the BufferedImage.
		System.out.println("nombre de lignes = " + nombre_lignes + "; nombre de colonnes = " + nombre_col);

		// Affichage de l'image
		try {
			showImage(img);
		} catch (IOException e) {
			e.printStackTrace(); // Prints this throwable and its backtrace to the standard error stream.
		}
	}
}
```

Exercice 0 (prise en main)

(2) Afficher la valeur du pixel qui se trouve dans le centre du quart supérieur droit

```
import java.io.*; // java.io.File, java.io.IOException
import javax.swing.*; // javax.swing.ImageIcon, javax.swing.JFrame, javax.swing.JLabel
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
public class Exercice0b {
        public static void showImage(BufferedImage bufferedImage) throws IOException {
              JFrame frame = new JFrame("Exercice0b");
              ImageIcon imageIcone = new ImageIcon(bufferedImage);
              JLabel jLabel = new JLabel(imageIcone);
              frame.getContentPane().add(jLabel);
              frame.pack();
              frame.setVisible(true);
         }

        public static void main(String[] args) {
                File path = new File("Test_Images" + File.separator + "doc1.jpg");
                BufferedImage img = null;

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

                int nombre_col  = img.getWidth();
                int nombre_lignes = img.getHeight();
                System.out.println("nombre de lignes = " + nombre_lignes + "; nombre de colonnes = " + nombre_col);

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

                int i = (nombre_lignes / 2) - (nombre_lignes / 4);
                int j = (nombre_col / 2) + (nombre_col / 4);

                int p = img.getRGB(j,i); // récupération des couleurs RGB du pixel a la position

                int a = (p>>24)&0xff;
        int r = (p>>16)&0xff;
        int g = (p>>8)&0xff; // p = 0xFF9A9BFA -> p>>B = 0xFAFF9A9B &0x000000ff = 0x98
        int b = p&0xff; // int b = p&0x000000ff;

        System.out.println("a = " +  a + " r = " +  r + " g = " + g + " b = " + b);

        // Affichage des marqueurs (optionnel)
        int[] rouge = {255, 0, 0, 255};
        for(int i2 = i - 2 ; i2 < i + 3 ; i2++) {
        for(int j2 = j - 2 ; j2 < j + 3 ; j2++) {
                img.getRaster().setPixel(j2,  i2, rouge);
        }
        }
        int[] blanc = {255, 255, 255, 255};
        img.getRaster().setPixel(j,  i, blanc);
        }
}
```

Exercice 1

```java
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.ImageIcon;
import java.io.IOException;
import java.awt.image.BufferedImage;
public class Exercice1 {
	static BufferedImage bresenham(BufferedImage img, int x1, int y1, int x2, int y2, int[] color) {
		int dy = y2 - y1;
		int dx = x2 - x1;
		int y = y1;
		double error = 0.0;
		double e_10  = (double) dy/dx;
		double e_01 = -1.0;

		for(int x = x1 ; x <= x2 ; x++) {
			// System.out.print("(" + x + "," + y + ")\n");
			img.getRaster().setPixel(x, y, color);
			error = error + e_10;

			if(error >= 0.5) {
				y++;
				error = error + e_01;
			}
		}

		return img;
	}

	public static void showImage(BufferedImage bufferedImage) throws IOException {
	      JFrame frame = new JFrame("Exercice1");

	      ImageIcon imageIcone = new ImageIcon(bufferedImage);
	      JLabel jLabel = new JLabel(imageIcone);

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

	public static void main(String[] args) {
		// A
		int width = 128;
		int height = 128;
		int[] noir = {0,0,0,255};
		BufferedImage img = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
		//WritableRaster raster = img.getRaster();
		for(int x = 0 ; x < width ; x++) {
			for(int y = 0 ; y < height ; y++) {
				img.getRaster().setPixel(x,  y, noir);
			}
		}

		// B
		int[] blanc = {255, 255, 255, 255};
		for(int x = 14 ; x < 114 ; x++) {
			img.getRaster().setPixel(x,  height/2, blanc);
		}

		// C
		int[] bleu = {0, 0, 255, 255};
		for(int x = 14 ; x < 114 ; x++) {
			img.getRaster().setPixel(x,  x, bleu);
		}
```

```
                // D
                int[] rouge = {255, 0, 0, 255};
                img = bresenham(img, 20, 50, 70, 70, rouge);

                // E
                int[] vert = {0, 255, 0, 255};
                int len = 100;
                img = bresenham(img, 10, 60, 10 + len,(int) ((int) 60 + len * Math.tan(Math.toRadians(30))), vert);

                // F
                int[] jaune = {255, 255, 0, 255};
                for(int x = 20 ; x < 120 ; x++) {
                        for(int y = 90 ; y < 110 ; y++) {
                                img.getRaster().setPixel(x,  y, jaune);
                        }
                }

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

                // G
                BufferedImage img2 = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
                for(int x = 0 ; x < width ; x++) {
                        for(int y = 0 ; y < height ; y++) {
                                int val_gris = (int) (y * 1.0 / height * 255);
                                int[] gris = {val_gris, val_gris, val_gris, 255};
                                img2.getRaster().setPixel(x, y, gris);
                        }
                }

                try {
                        showImage(img2);
                } catch(IOException e) {
                        e.printStackTrace();
                }
        }
}
```

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

Université de Paris

TD-TP 1 : Analyse et traitement d'image

Exercice 0 (prise en main)

- Charger une des images dans le dossier *Test_Images* et afficher là. Vous utiliserez la classe `java.awt.image.BufferedImage`
- Afficher la valeur du pixel qui se trouve dans le centre du quart supérieur droit.

Exercice 1

A. Créer une image noire de 128x128
B. Dessiner une ligne horizontale de 100 pixels de longueur
C. Dessiner une ligne oblique à 45°
D. Dessiner une ligne entre deux pixels dont on donnera les coordonnées
E. Dessiner une ligne oblique à 30°
F. Dessiner un rectangle sur la même image
G. Créer une image en dégradés de niveaux de gris

https://www.sylvainlobry.com/l3images/