University Echahid Hama Lakhdar, El-oued
Institute of Exact Sciences
Department of Computer Science
$2^{ed} Master$:
Artificial Intelligence and Data Science
Semester : 3.2025

Lab1 : How Computers See Images

Master Level – Computer Vision

1 Image representation

Objective

Understand how computers read, represent, and preprocess images using Python, OpenCV, and Matplotlib.

Required Libraries

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

1. Reading and Displaying Images

Example Image

Suppose we have an image file named `car.jpg`, which contains a color photograph of a car. This file will be used as an example throughout the lab to demonstrate how digital images are represented and processed using the `OpenCV` and `Matplotlib` libraries in Python.

```
# Read the image
image = cv2.imread('car.jpg')

# Display using Matplotlib
plt.imshow(image)
plt.title("Original (BGR) image")
plt.show()
```

Listing 1 – Read and display image

Q1.

— Explain why the image colors look distorted when using `plt.imshow()` directly after `cv2.imread()`.

— Print the shape of the image array. What do each of the three dimensions represent ?

```
print(image.shape)
```

2. Converting BGR to RGB

```python
# Convert BGR to RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

plt.imshow(image_rgb)
plt.title("RGB Image")
plt.show()
```

Listing 2 – Convert to RGB

Q2.

— Compare the visual result with the previous one.

— Why does OpenCV use BGR while Matplotlib uses RGB ?

3. Normalizing Image Intensities

```python
# Normalize pixel values to [0, 1]
normalized = image_rgb.astype('float32') / 255.0

plt.imshow(normalized)
plt.title("Normalized RGB Image")
plt.show()

print("Pixel range before normalization:", image_rgb.min(), "   ", image_rgb.
    max())
print("Pixel range after normalization:", normalized.min(), "   ", normalized.
    max())
```

Listing 3 – Normalize pixel values

Q3.

— Why is normalization important before feeding images into deep learning models ?

— Modify the code to perform z-score normalization :

```python
zscore = (image_rgb - np.mean(image_rgb)) / np.std(image_rgb)
plt.imshow(np.clip(zscore, 0, 1))
```

— Discuss how the visualization changes.

4. Converting to Grayscale

```python
# Convert RGB to Grayscale
gray = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2GRAY)

plt.imshow(gray, cmap='gray')
plt.title("Grayscale Image")
plt.show()
```

Listing 4 – Convert RGB to Grayscale

Q4.

— Compute and print the shape of the grayscale image.

— Explain how grayscale values are computed from RGB channels.

— What kind of information is lost after grayscale conversion ?

5. Visualizing Image Histograms

```
# RGB histograms
colors = ('r', 'g', 'b')
for i, col in enumerate(colors):
    hist = cv2.calcHist([image_rgb], [i], None, [256], [0, 256])
    plt.plot(hist, color=col)
plt.title("RGB Histograms")
plt.show()

# Grayscale histogram
plt.hist(gray.ravel(), bins=256, range=[0,256], color='gray')
plt.title("Grayscale Histogram")
plt.show()
```

Listing 5 – Plot histograms

Q5.

— Compare RGB and grayscale histograms.

— What do you infer about color richness and brightness ?

6. Experiment : Your Own Image

Q6.

— Replace `'car.jpg'` with your own image.

— Repeat all steps and describe the observed differences.

7. Reflection and Extension

Q7.

— If you feed the RGB image into a CNN, its input size is $(H, W, 3)$. How does this differ for grayscale images ?

— How does this change affect model parameters and computational cost ?