SI100B Project Face Detection and emotion classification

Lecture 1

Objectives

- Setup a development environment.
- Familiarize yourself with the basic knowledge of digital images.

1 Environment Setup

We use VS Code as the default editor in this course.
The textbook is written with VS Code and Python virtual environments in mind, but you're welcome to use any other editor you're comfortable with.

1. Download VS Code. If your are not have it yet.
2. Make a folder for our project. It's very important to establish a good file organization structure right from the first class. Based on past experience, many students tend to forget what they did earlier and how they did it—this can seriously hinder your ability to integrate everything into a final pipeline and write your report. Below is my file structure: I created a folder named **SI100BProject**, and I put the materials for each class into a separate subfolder. This way, the work for each session stays clear and well organized.

```
PS C:\Users\zhenghao\Documents\SI00BProject> ls .


    目录: C:\Users\zhenghao\Documents\SI00BProject


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2025/12/1     15:03                LAB1
d-----         2025/12/1     15:03                LAB2
d-----         2025/12/1     15:03                LAB3
d-----         2025/12/1     15:03                LAB4
d-----         2025/12/1     15:03                LAB5
d-----         2025/12/1     15:03                LAB6
d-----         2025/12/1     15:03                LAB7
```

Figure 1: Power Shell perspective

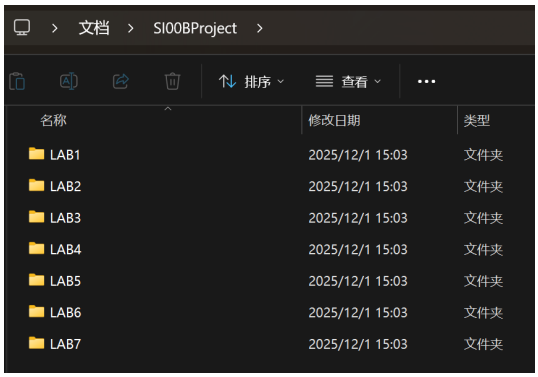


Figure 2: File explorer perspective

3. Open this folder in VS Code.

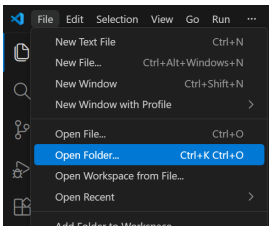


Figure 3: Open folder as workspace

4. Open a terminal and set it as a VS Code workspace by typing ”code .” in the terminal.

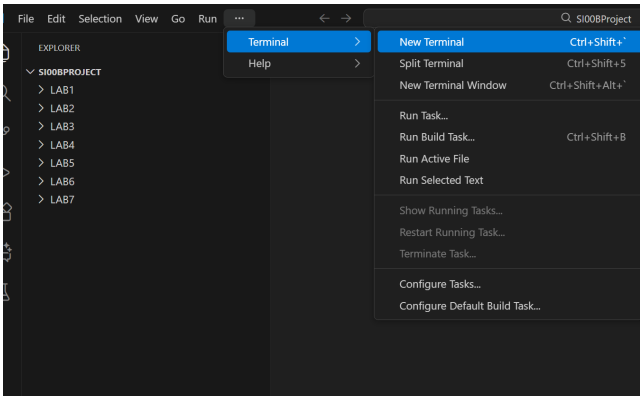


Figure 4: How to open terminal

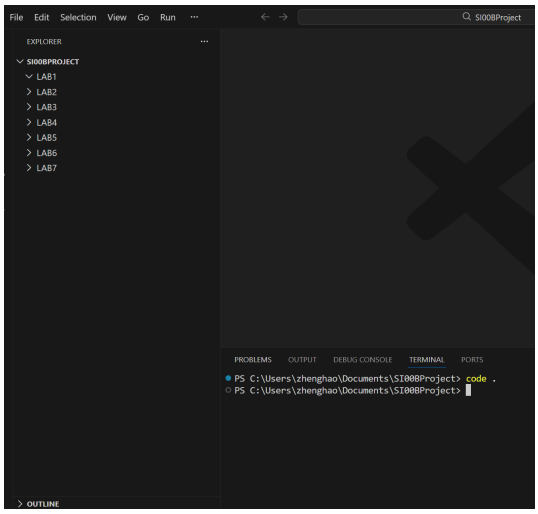
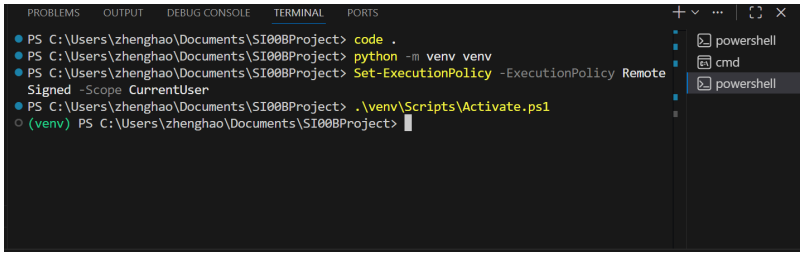


Figure 5: Typing code

5. Create a virtual environment in this workspace and activate it. Once you setup this virtual environment, it will all use this environment every time you open this workspace.

```
python -m venv venv
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
.\venv\Scripts\Activate.ps1
```



6. Copy the requirements file and install prerequisites.

```
pip install -r requirements.txt
```

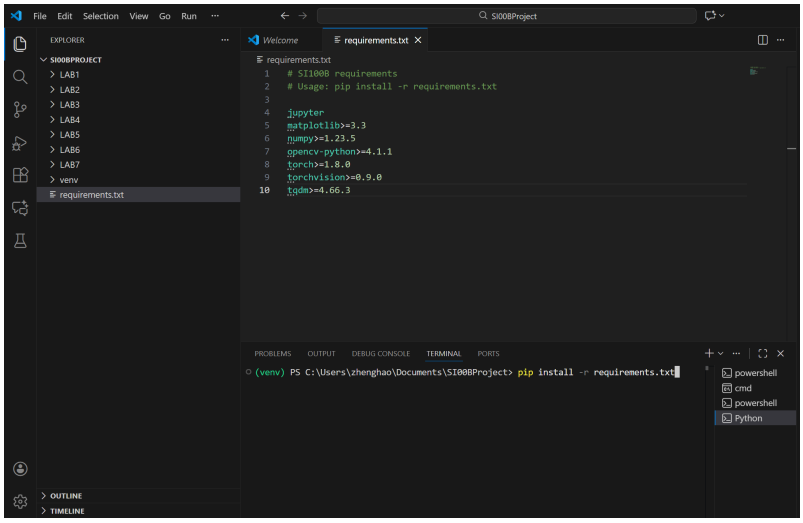


Figure 6: Install requirements

7. Open a test file and Test if the installation is successful. No news is good news—if you don't see any error messages, congratulations, you've successfully set up your environment!

```
import cv2
import numpy as np
```

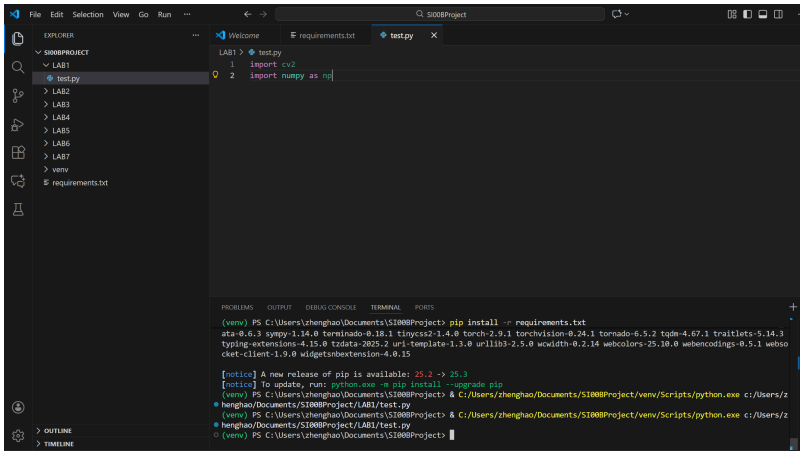


Figure 7: Test

2 Fundamentals of Digital Imaging

How digital images are stored on computers?

2.1 Color

Please take a close look at the following picture 8.

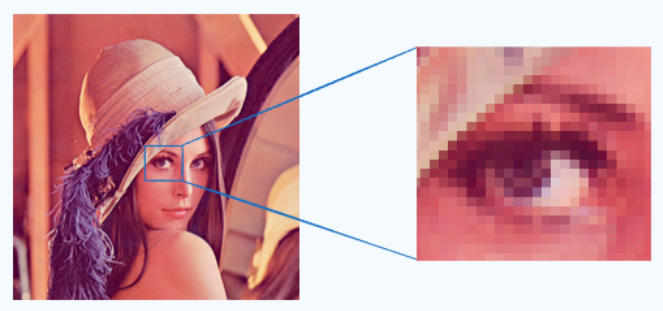


Figure 8: lenna image

When we zoom in the image, we can see lots of cells which we call it pixel. A pixel represents a single point in an image. It is the building block of all digital images, analogous to a tiny dot of color on a printed photograph.
Each pixel in an image typically has a specific color, which is defined by its color channels. For an RGB (Red, Green, Blue) image, each pixel is represented by three values corresponding to the intensity of each color channel. These values usually range from 0 to 255, allowing for over 16 million possible color combinations (256 red x 256 green x 256 blue).

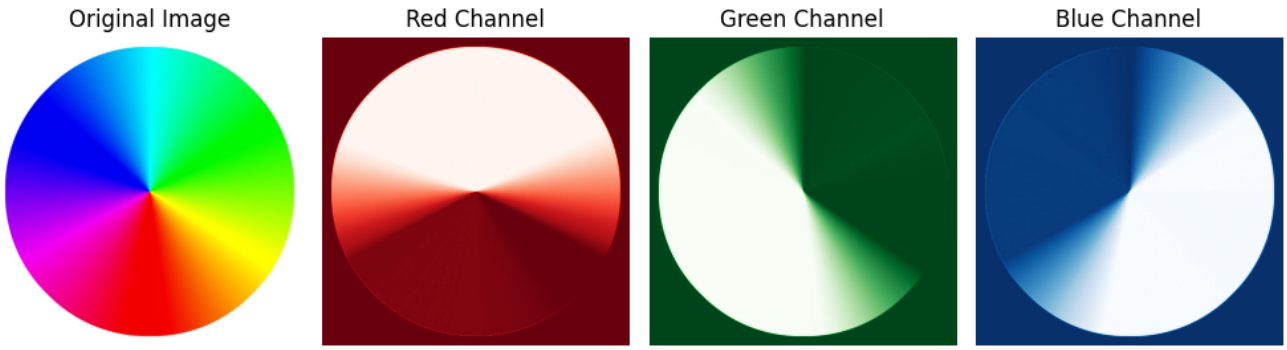


Figure 9: Split channels

2.2 Coordinate System

The image coordinate system is based on a 2D Cartesian grid.

- The origin (0, 0) is at the top-left corner of the image.
- The X-axis increases to the right.

- The Y-axis increases downward.

Each pixel in an image is defined by a pair of coordinates (X, Y), where:

- X is the horizontal position.
- Y is the vertical position.

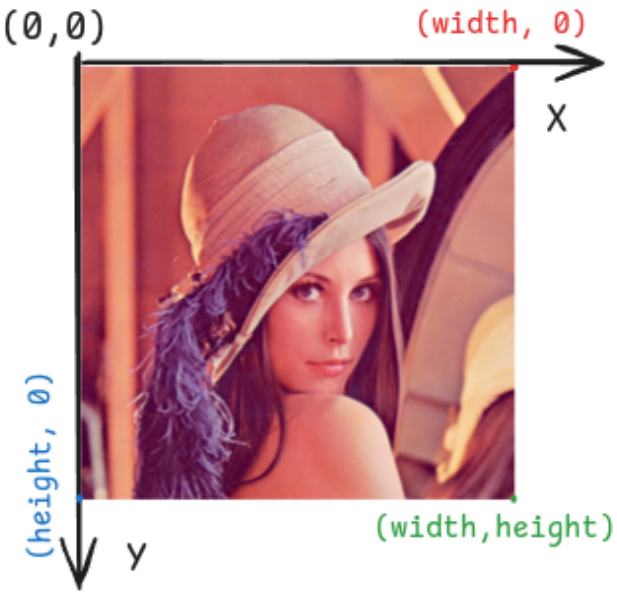


Figure 10: Coordinate System

Now we can give the mathematical representation. A colorful digital image can be represented as three 2D arrays. Each element of the 2D-array corresponds to a pixel. Every pixel can be accessed by it's coordinate values.

$$R = I_R(x, y), G = I_G(x, y), B = I_B(x, y) \tag{1}$$

where $R, G, B \in [0, 255]$

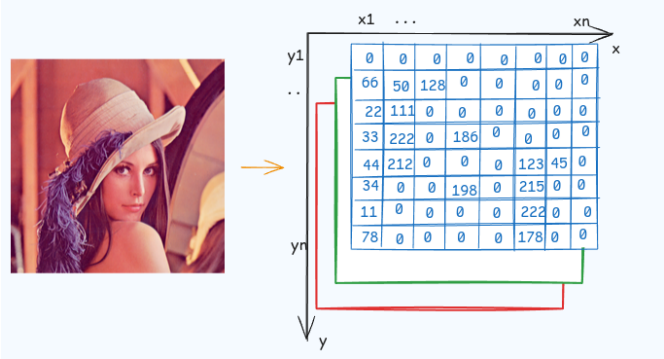


Figure 11: Image resprentation

Use this link PixSpy to test your image.

3 Exmaple

Read the following piece of code, think about what kind of image it will output, and then run it to see if it matches your expectations

```python
import cv2
import numpy as np
if __name__ == '__main__':
    # Generator an empty image with 200 height and 300 width
    blanks = np.zeros((200, 300, 3), dtype=np.uint8)

    ###FILL red when x in range [100, 200] and y in range [50, 150]
    for i in range(50, 150):
        for j in range(100, 200):
            blanks[i, j] = np.array([0, 0, 255])

    cv2.imwrite("create.png", blanks)
    cv2.imshow("src", blanks)
    cv2.waitKey(0)
```

4 Test

Fill the blanks.

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

if __name__ == '__main__':
    # Generator an empty image with 480 height and 640 width

    ###FILL red when x in range [0, 160] and y in range [0, 120]

    ###FILL GREEN when x in range [160, 320] and y in range [120, 240]

    ###FILL GREEN when x in range [320, 480] and y in range [240, 360]

    ###FILL GRAY[128, 128, 128] when x in range [480, 640] and y in range [360, 480]

    cv2.imwrite("create.png", blanks)
    cv2.imshow("src", blanks)
    cv2.waitKey(0)
```