Digital Image Filtering

CST 205

Digital Image Processing

- Application of various algorithms on image data
- Example: Image smoothing
  - Reducing **noise** from the image.
    - **Noise:** random changes in brightness and color levels within the image data.

“Salt and Pepper” Image Noise

Image Smoothing

- A good filtering/enhancement method should:
  - remove image noise
  - maintain edge information

Fundamentals of Color Imaging

- **Tristimulus theory** of color representation
  - The human retina has 3 kinds of color sensors — **cones**
    - red, green, blue are in the peak response range of each of the cones
- Each pixel in a Red-Green-Blue image can be designated as a 3-tuple of red, green, and blue **intensity values.**
  - e.g., (173,101, 95)

Pixel coordinates

- With an imaging library, we can use pixel coordinates to add lines, shapes, and text to the images.

- The pixel coordinate system starts in the upper-left corner.

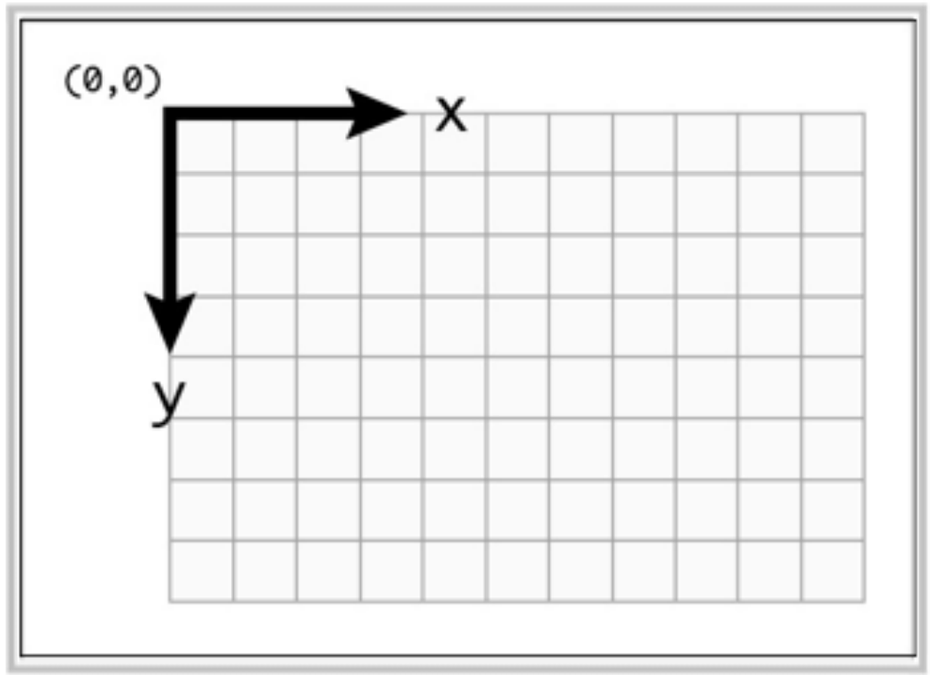

Image processing with Pillow

- A bit of history: Python Imaging Library (PIL) used to be the standard solution for manipulating images using Python

- Development of PIL ceased in 2011. Pillow forked the project and continued it.

- Pillow docs available here

Pillow installation

- With your virtual environment activated:
  - `pip install Pillow`
  - (See Lab 4 for more details on virtual environments)

Pillow fundamentals

- Pillow is based on an `Image` class that can be opened and saved.

- Our first step is to create a so-called Pillow Image object. With this object we can then:

  - get information about the underlying image

  - apply various operations to the image

Note: We will talk about object-oriented Python in another lecture. For now, you can think of the `Image` object as a way we can refer to an image and perform operations on it.

Code sample

```
from PIL import Image   # an error here means that Pillow is not installed
im = Image.open('images/chualar_sign.jpg')
# print(im)

print(im.size) # (1024, 768)

width, height = im.size
```

Pixel access

- How can we access all of the pixels in our image?
- Pillow provides several approaches. For now, we will use a nested loop.
    - We can write a nested loop such that we travel down each column.

Small example

- Assume we have a picture with a width of 4 pixels and a height of 5 pixels.

```
pic_width = 4
pic_height = 5

for x in range(pic_width):          # loop from 0 to pic_width - 1
    for y in range(pic_height):     # loop from 0 to pic_height - 1
        print((x,y))
```

- This prints out a list of coordinate values going down each column.

Pillow's `getpixel()` method

- If we have a Pillow Image object, we can use a method called getpixel(), described here

```
from PIL import Image
im  = Image.open('images/chualar_sign.jpg')

width, height = im.size

big_pixel_list = []

for x in range(width):
    for y in range(height):
        big_pixel_list.append(im.getpixel((x,y)))
```

Get to know our image better

```
# how many pixels in the image
print(len(big_pixel_list))

# let's take a look at one of the pixels
print(big_pixel_list[100])
```

Pillow's `putpixel()` method

- Described here

```
import secrets

for x in range(width):
  for y in range(height):
    rgb_val = (
      secrets.choice(range(40,200)),
      secrets.choice(range(150,170)),
      secrets.choice(range(50,200))
      )
    im.putpixel((x,y), rgb_val)

im.save("images/new_chualar.jpg")
```

Creating a new empty `Image` object

- Pillow's `new()` method allows us to create a new image with defined width and height and mode.
  - Documentation here
- More information on Pillow's color modes here

```
# 8-bit grayscale image with width 400 and height 500

my_image = Image.new('L', (400,500))
```