7 - 8.4. Scaling

November 18, 2024

Jarrian Vince G. Gojar
Instructor I
College of Information and Communications Technology, Sorsogon State University, Philippines

1 Introduction

`Scaling` in image processing refers to the resizing of an image. Scaling is used to enlarge or reduce the size of an image. Scaling an image means changing the dimensions of the image. The dimensions of the image can be changed by using the scaling factor. The scaling factor is a ratio of the new dimension to the original dimension. The scaling factor can be greater than 1 or less than 1. If the scaling factor is greater than 1, the image is enlarged. If the scaling factor is less than 1, the image is reduced.

There are two types of scaling:

1. **Uniform Scaling**: In uniform scaling, the scaling factor is the same in both the x and y directions.
2. **Non-Uniform Scaling**: In non-uniform scaling, the scaling factor is different in the x and y directions.

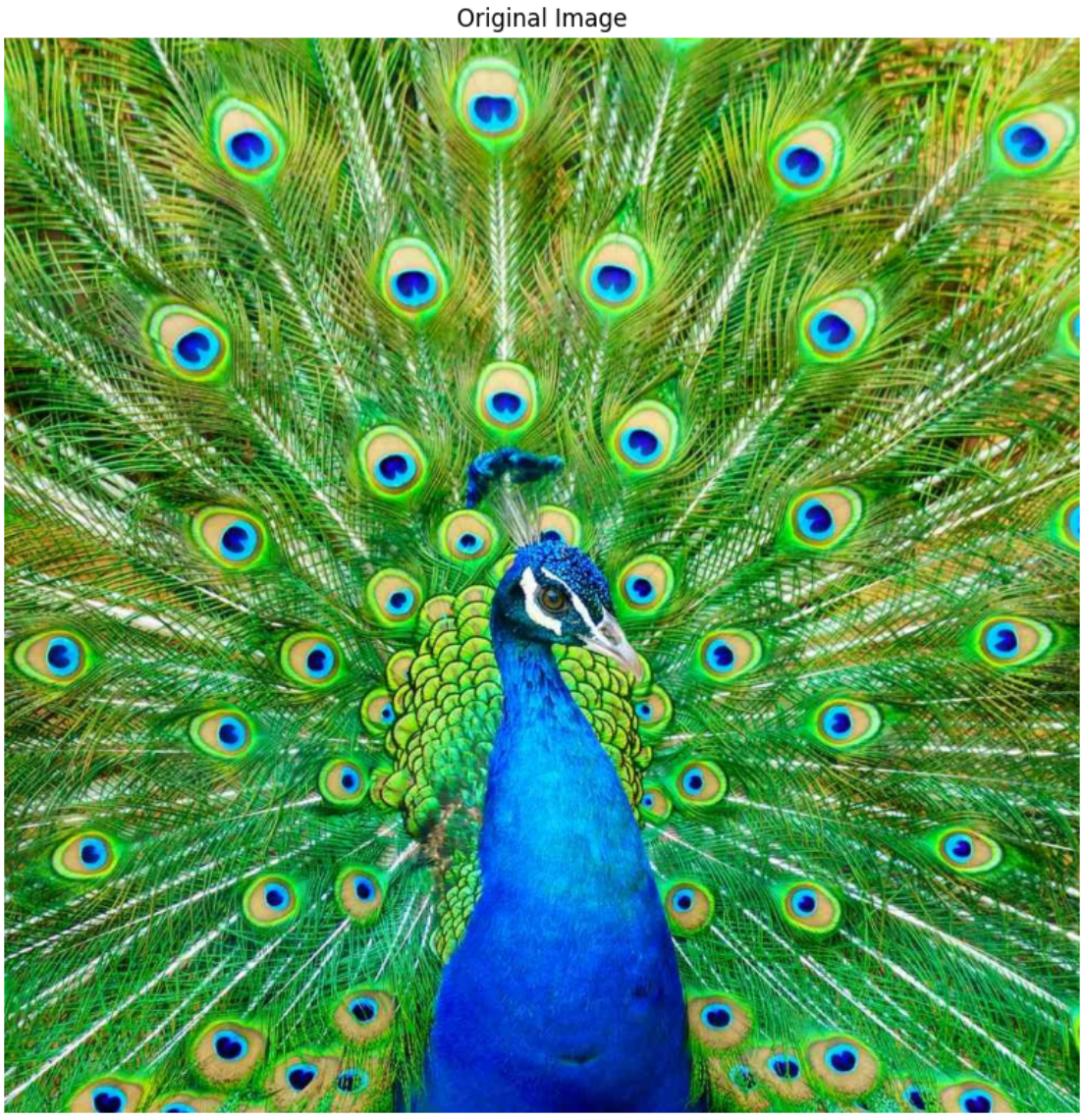
2 Setup

[ ]:
```
%pip install opencv-python opencv-contrib-python numpy matplotlib scipy
```

3 Initial Setup

[1]:
```
# Import Libraries
import os
import cv2
import matplotlib.pyplot as plt

# Asset Root
asset_root = os.path.join(os.getcwd(), '../../assets')

# Image Path
image_path = os.path.join(asset_root, 'images', 'peacock.jpg')
```

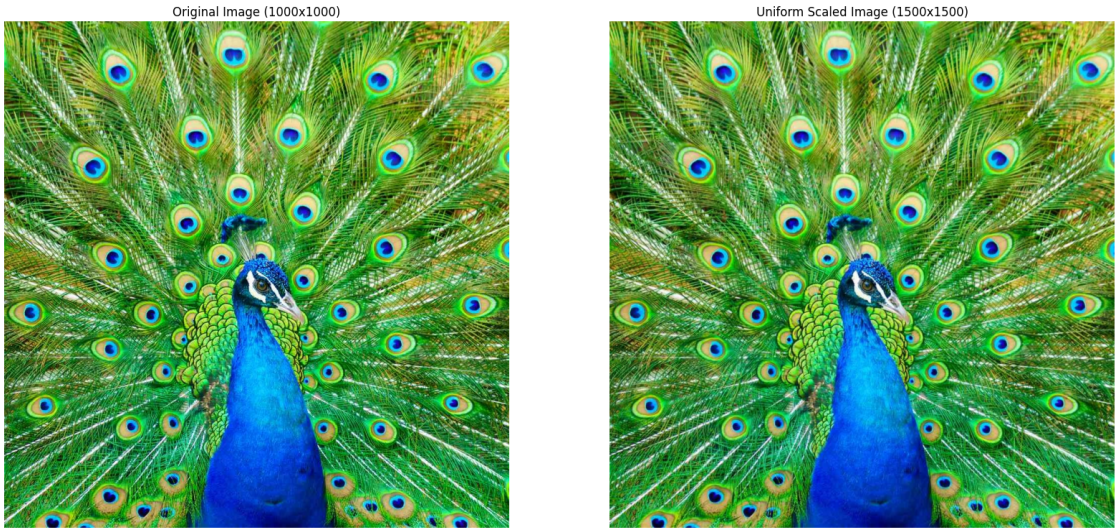
```
# Read Image and convert to RGB
input_image = cv2.cvtColor(cv2.imread(image_path), cv2.COLOR_BGR2RGB)

# Display Both Image
plt.figure("Uniform Scaling", figsize=(10, 10))

plt.imshow(input_image, cmap='gray')
plt.title("Original Image")
plt.axis('off')

plt.show()
```



3.1 Uniform Scaling

In `uniform scaling`, the scaling factor is the same in both the x and y directions. The scaling factor can be greater than 1 or less than 1. If the scaling factor is greater than 1, the image is enlarged. If the scaling factor is less than 1, the image is reduced. The uniform scaling can be done using the `cv2.resize()` function of the OpenCV library.

[2]:
```
# Uniform Scaling
def uniform_scaling(image, scaling_factor):
    # Get Image Shape
    h, w, _ = image.shape

    # Calculate New Height and Width
    new_h = int(h * scaling_factor)
    new_w = int(w * scaling_factor)

    # Resize Image
    scaled_image = cv2.resize(image, (new_w, new_h), interpolation=cv2.
 ↪INTER_LINEAR)

    return scaled_image

# Scaling Factor
scaling_factor = 1.5

# Uniform Scaling
uniform_scaled_image = uniform_scaling(input_image, scaling_factor)

# Display Both Image
plt.figure("Uniform Scaling", figsize=(20, 10))

plt.subplot(1, 2, 1)
plt.imshow(input_image)
plt.title(f"Original Image ({input_image.shape[1]}x{input_image.shape[0]})")
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(uniform_scaled_image)
plt.title(f"Uniform Scaled Image ({uniform_scaled_image.
 ↪shape[1]}x{uniform_scaled_image.shape[0]})")
plt.axis('off')

plt.show()
```



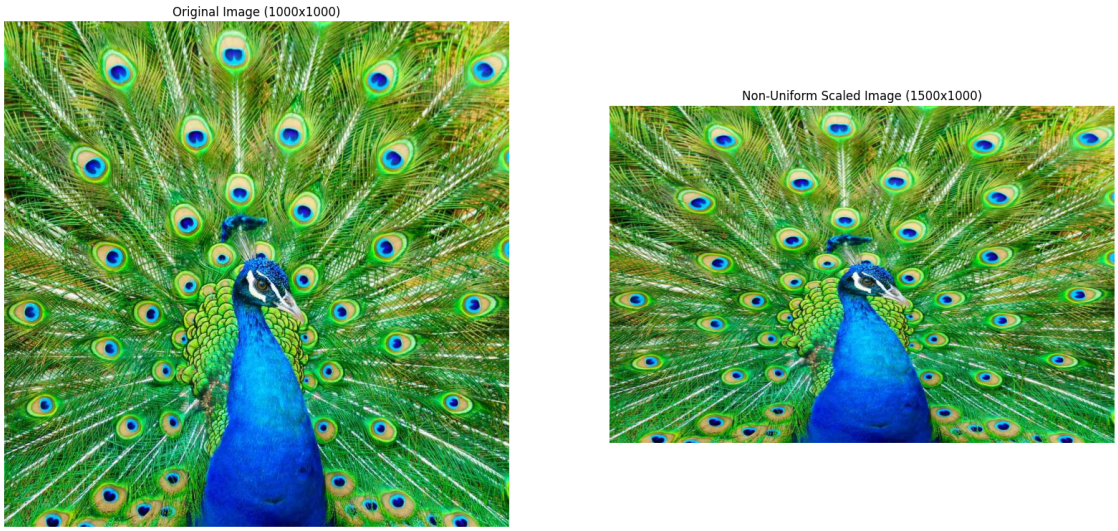
The above code performs uniform scaling on the input image. The scaling factor is 1.5. The `uniform_scaling()` function takes the input image and the scaling factor as input and returns the scaled image. The `cv2.resize()` function is used to resize the image. The `cv2.resize()` function takes the input image and the size of the output image as input. The size of the output image is calculated by multiplying the height and width of the input image by the scaling factor. The `cv2.resize()` function returns the scaled image.

3.2 Non-Uniform Scaling

In `non-uniform scaling`, the scaling factor is different in the x and y directions. This results in the image being stretched or compressed in one direction. For example, if the scaling factor in the x direction is 1.5 and the scaling factor in the y direction is 1.0, the image will be stretched in the x direction by a factor of 1.5. The aspect ratio of the image is not preserved in non-uniform scaling.

[3]:
```python
# Non-Uniform Scaling
def non_uniform_scaling(image, scaling_factor_width, scaling_factor_height):
    # Get Image Shape
    h, w, _ = image.shape

    # Calculate New Height and Width
    new_h = int(h * scaling_factor_height)
    new_w = int(w * scaling_factor_width)

    # Resize Image
    scaled_image = cv2.resize(image, (new_w, new_h), interpolation=cv2.
 ↪INTER_LINEAR)

    return scaled_image

# Scaling Factor
```

```
scaling_factor_width = 1.5
scaling_factor_height = 1.0

# Non-Uniform Scaling
non_uniform_scaled_image = non_uniform_scaling(input_image,␣
 ↪scaling_factor_width, scaling_factor_height)

# Display Both Image
plt.figure("Non-Uniform Scaling", figsize=(20, 10))

plt.subplot(1, 2, 1)
plt.imshow(input_image)
plt.title(f"Original Image ({input_image.shape[1]}x{input_image.shape[0]})")
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(non_uniform_scaled_image)
plt.title(f"Non-Uniform Scaled Image ({non_uniform_scaled_image.
 ↪shape[1]}x{non_uniform_scaled_image.shape[0]})")
plt.axis('off')

plt.show()
```



The above code performs non-uniform scaling on the input image. The scaling factor in the x-direction is 1.5 and the scaling factor in the y-direction is 1.0. The `non_uniform_scaling()` function takes the input image and the scaling factors in the x and y directions as input and returns the scaled image. The `cv2.resize()` function is used to resize the image. The `cv2.resize()` function takes the input image and the size of the output image as input. The size of the output image is calculated by multiplying the height and width of the input image by the scaling factors

in the x and y directions. The `cv2.resize()` function returns the scaled image.

4 Summary

- `Scaling` in image processing refers to the resizing of an image. Scaling is used to enlarge or reduce the size of an image.
- There are two types of scaling: `Uniform Scaling` and `Non-Uniform Scaling`.
- In `Uniform Scaling`, the scaling factor is the same in both the x and y directions.
- In `Non-Uniform Scaling`, the scaling factor is different in the x and y directions.
- The `cv2.resize()` function of the OpenCV library is used to perform scaling on an image.
- The size of the output image is calculated by multiplying the height and width of the input image by the scaling factor.

5 References

- Thomas G. (2022). Graphic Designing: A Step-by-Step Guide (Advanced). Larsen & Keller. ISBN: 978-1-64172-536-1
- Singh M. (2022). Computer Graphics and Multimedia. Random Publications LLP. ISBN: 978-93-93884-95-4
- Singh M. (2022). Computer Graphics Science. Random Publications LLP. ISBN: 978-93-93884-03-9
- Singh M. (2022). Computer Graphics Software. Random Publications LLP. ISBN: 9789393884114
- Tyagi, V. (2021). Understanding Digital Image Processing. CRC Press.
- Ikeuchi, K. (Ed.). (2021). Computer Vision: A Reference Guide (2nd ed.). Springer.
- Bhuyan, M. K. (2020). Computer Vision and Image Processing. CRC Press.
- Howse, J., & Minichino, J. (2020). Learning OpenCV 4 Computer Vision with Python 3: Get to grips with tools, techniques, and algorithms for computer vision and machine learning. Packt Publishing Ltd.
- Kinser, J. M. (2019). Image Operators: Image Processing in Python. CRC Press.