Homework 6

(1) Problem statement

Image Enlargement

Step 1: Choose a grayscale image I

Step 2: Zero interleave

$$I'(i,j)=\begin{cases} I((i+1)/2,(j+1)/2) & \text{if } i,j: \text{ odd} \\ 0 & \text{otherwise} \end{cases}$$

Step 3: Fill values by convolving I' with

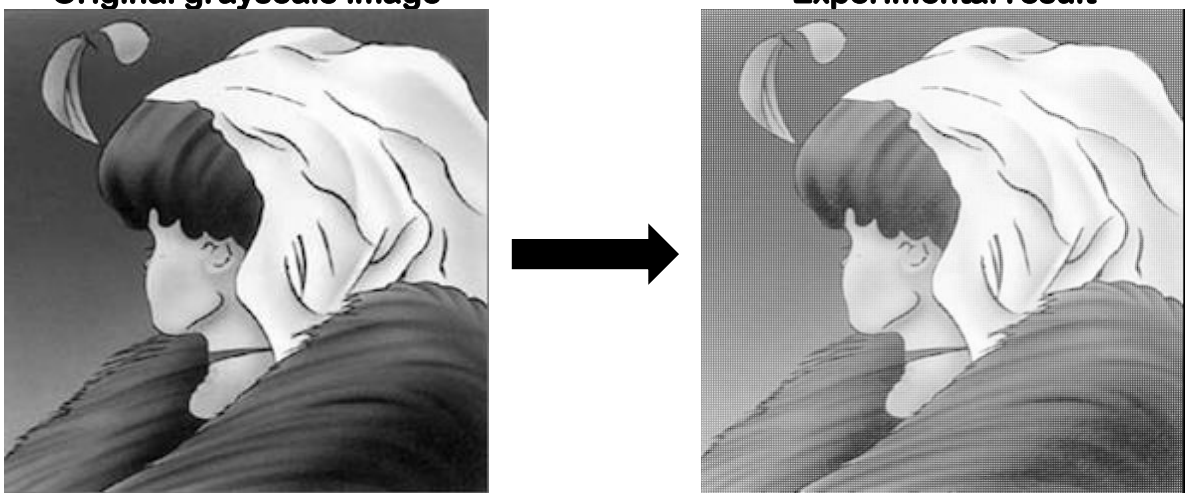
(i) $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ NN interpolation (ii) $\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ Bilinear interpolation

Step 4: Output enlarged images

(2.1) Experimental results

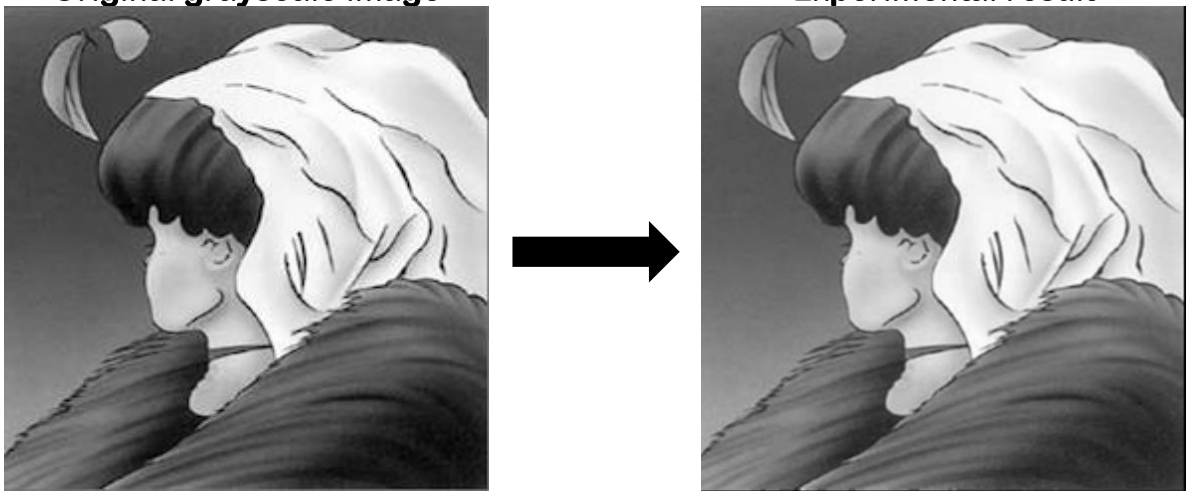
(i) NN interpolation

Original grayscale image → **Experimental result**



(ii) Bilinear interpolation

Original grayscale image → **Experimental result**



(2.2) Source code

(i) Implement Image Enlargement & filling use NN interpolation

```python
# HW6-(i) (Implement Image Enlargement & filling use NN interpolation)

from PIL import Image
import numpy as np

# Step1 : Input a grayscale image I
I = Image.open("W.E.Hill.png").convert("L")
width, height = I.size
new_row = width * 2
new_column = height * 2

# Step2 : Zero interleave
enlarged_image = I
enlarged_image = enlarged_image.resize((new_row, new_column))
I2 = np.array(enlarged_image)
for j in range(I.size[0] * 2):
    for i in range(I.size[1] * 2):
        if (i % 2 == 0) and (j % 2 == 0):  # I:even
            I2[j][i] = I.getpixel((i / 2, j / 2))
        else:                              # I:odd
            I2[j][i] = 0

# Step3 : Filling(NN interpolation)
NN_mask = [[1, 1, 3],
           [1, 1, 0],
           [0, 0, 0]]
# Convolution I2 with NN interpolation
for j in range(len(I2[1])):
    for i in range(len(I2[0])):
        tmp = 0
        for b in range(len(NN_mask)):
            for a in range(len(NN_mask)):
                if (j+b-1 >= new_column) or (i+a-1 >= new_row):
                    tmp += 0 * NN_mask[b][a]
                else:
                    tmp += I2[j+b-1][i+a-1] * NN_mask[b][a]
        # reDraw image
        enlarged_image.putpixel((i, j), int(tmp))

# Step4 : Save and output enlarged images
enlarged_image.save("enlarged_image.png")
enlarged_image.show()

```

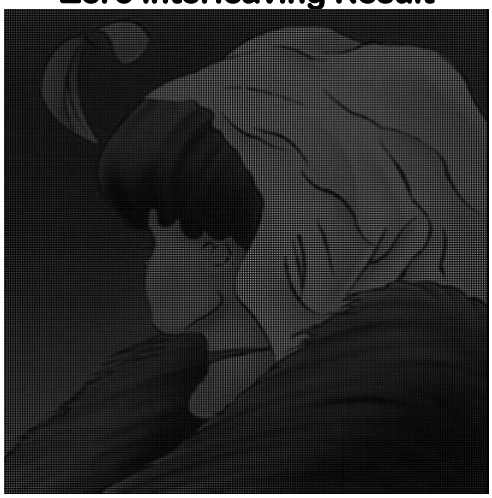
(ii) Implement Image Enlargement & filling use Bilinear interpolation

```python
# HW6-(ii) (Implement Image Enlargement & filling use Bilinear interpolation)

from PIL import Image
import numpy as np

# Step1 : Input a grayscale image I
I = Image.open("W.E.Hill.png").convert("L")
width, height = I.size
new_row = width * 2
new_column = height * 2

# Step2 : Zero interleave
enlarged_image = I
enlarged_image = enlarged_image.resize((new_row, new_column))
I2 = np.array(enlarged_image)
for j in range(I.size[0] * 2):
    for i in range(I.size[1] * 2):
        if (i % 2 == 0) and (j % 2 == 0):  # I:even
            I2[j][i] = I.getpixel((i / 2, j / 2))
        else:                              # I:odd
            I2[j][i] = 0

# Step3 : Filling(Bilinear interpolation)
Bilinear_mask = [[1, 2, 1],
                 [2, 4, 2],
                 [1, 2, 1]]
# Convolution I2 with NN interpolation
for j in range(len(I2[1])):
    for i in range(len(I2[0])):
        tmp = 0
        for b in range(len(Bilinear_mask)):
            for a in range(len(Bilinear_mask)):
                if (j+b-1 >= new_column) or (i+a-1 >= new_row):
                    tmp += 0 * Bilinear_mask[b][a]
                else:
                    tmp += I2[j+b-1][i+a-1] * Bilinear_mask[b][a]
        # reDraw image
        enlarged_image.putpixel((i, j), int(tmp/4))

# Step4 : Save and output enlarged images
enlarged_image.save("enlarged_image.png")
enlarged_image.show()

```

(2.3) Comments

Original grayscale image

Zero interleaving Result



NN interpolation Result

Bilinear interpolation Result