Digital Image Porcessing Laboratory:

Image Halftoning

Alexandre Olivé Pellicer

1. **Introduction**

2. **Image Fidelity Metrics**

3. **Thresholding and Random Noise Binarization**

    3.1. Hand in the original image and the result of thresholding

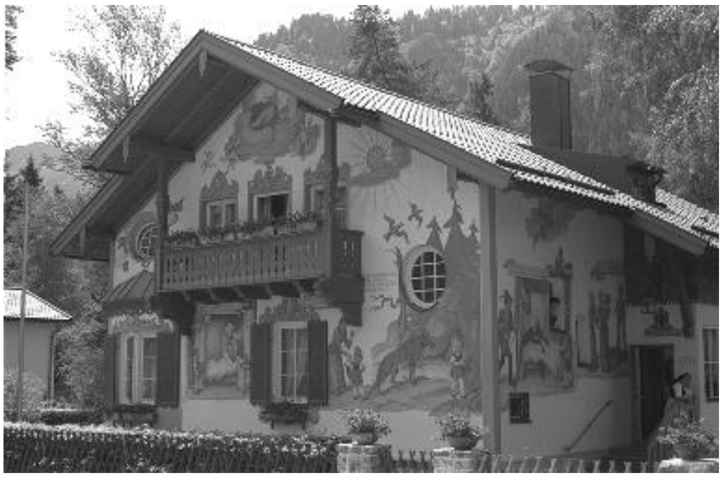


Fig 1: Original image

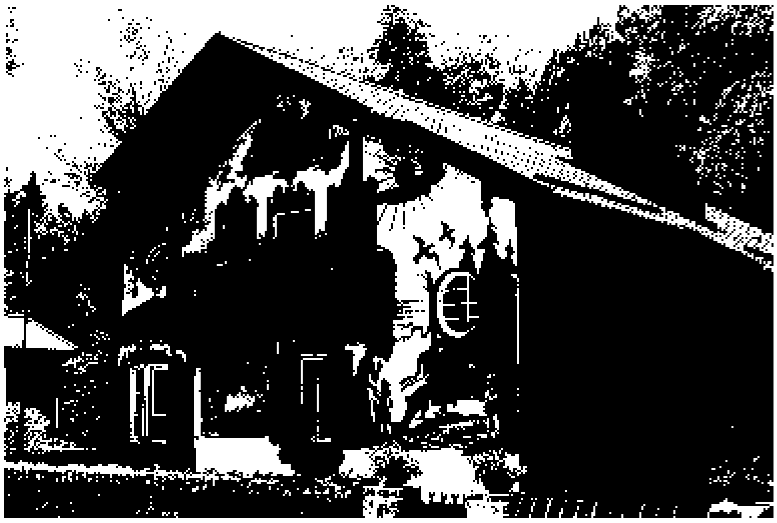


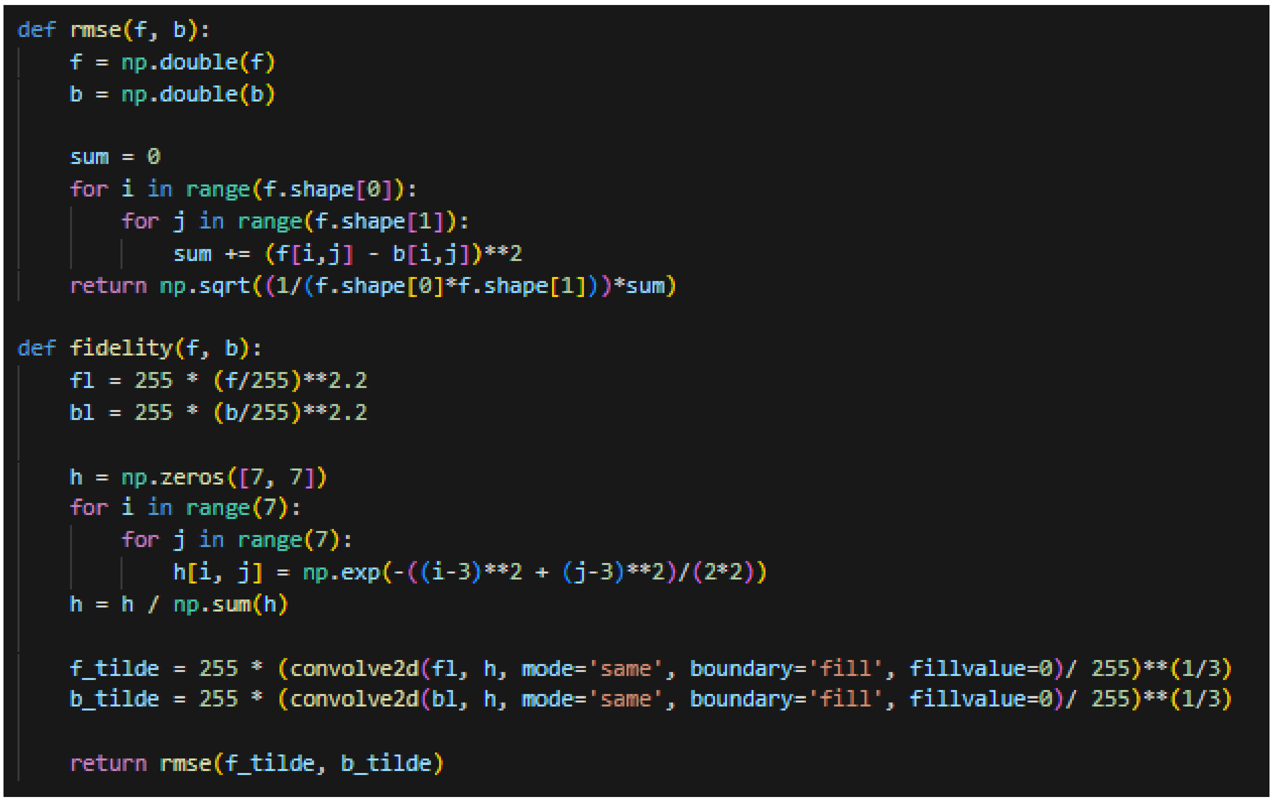
Fig 2: Thresholding image

3.2. Submit the computed RMSE and fidelity values

RMSE: 87.3933

Fidelity: 77.3371

3.3. Hand in the code for your fidelity function



```
def rmse(f, b):
    f = np.double(f)
    b = np.double(b)

    sum = 0
    for i in range(f.shape[0]):
        for j in range(f.shape[1]):
            sum += (f[i,j] - b[i,j])**2
    return np.sqrt((1/(f.shape[0]*f.shape[1]))*sum)

def fidelity(f, b):
    fl = 255 * (f/255)**2.2
    bl = 255 * (b/255)**2.2

    h = np.zeros([7, 7])
    for i in range(7):
        for j in range(7):
            h[i, j] = np.exp(-((i-3)**2 + (j-3)**2)/(2*2))
    h = h / np.sum(h)

    f_tilde = 255 * (convolve2d(fl, h, mode='same', boundary='fill', fillvalue=0)/ 255)**(1/3)
    b_tilde = 255 * (convolve2d(bl, h, mode='same', boundary='fill', fillvalue=0)/ 255)**(1/3)

    return rmse(f_tilde, b_tilde)
```

4. Ordered Dithering

4.1. The three Bayer index matrices of sizes 2 × 2, 4 × 4, and 8 × 8

Bayer index 2*2:

$$I = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$$

Bayer index 4*4:

$$I = \begin{bmatrix} 5 & 9 & 6 & 10 \\ 13 & 1 & 14 & 2 \\ 7 & 11 & 4 & 8 \\ 15 & 3 & 12 & 0 \end{bmatrix}$$

Bayer index 8*8:

$$I = \begin{bmatrix} 21 & 37 & 25 & 41 & 22 & 38 & 26 & 42 \\ 53 & 5 & 57 & 9 & 54 & 6 & 58 & 10 \\ 29 & 45 & 17 & 33 & 30 & 46 & 18 & 34 \\ 61 & 13 & 49 & 1 & 62 & 14 & 50 & 2 \\ 23 & 39 & 27 & 43 & 20 & 36 & 24 & 40 \\ 55 & 7 & 59 & 11 & 52 & 4 & 56 & 8 \\ 31 & 47 & 19 & 35 & 28 & 44 & 16 & 32 \\ 63 & 15 & 51 & 3 & 60 & 12 & 48 & 0 \end{bmatrix}$$

4.2. The three halftoned images produced by the three dither patterns

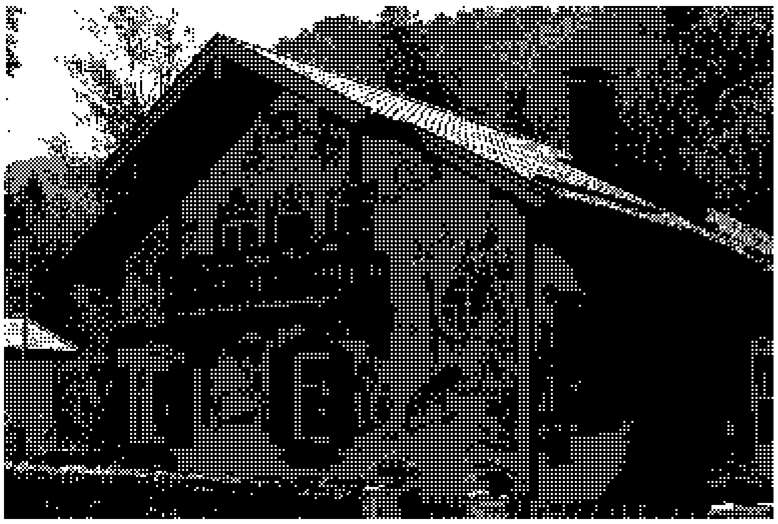


Fig 3: The halftone image produced by Bayer dithering of size 2*2

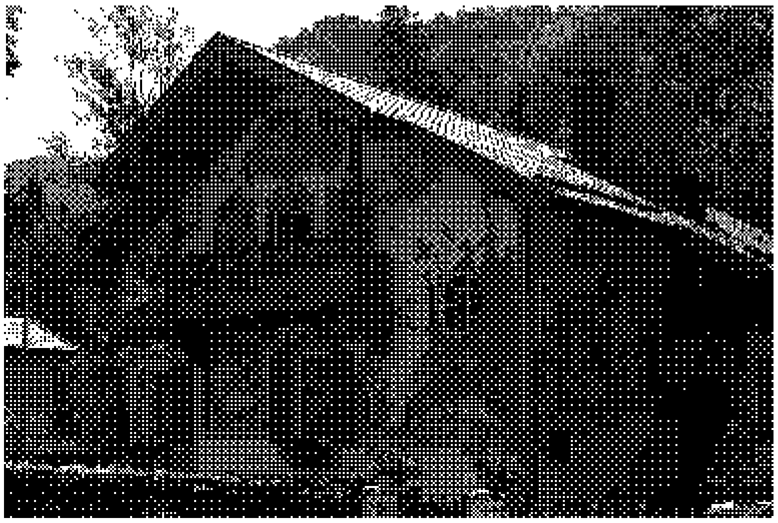


Fig 4: The halftone image produced by Bayer dithering of size 4*4

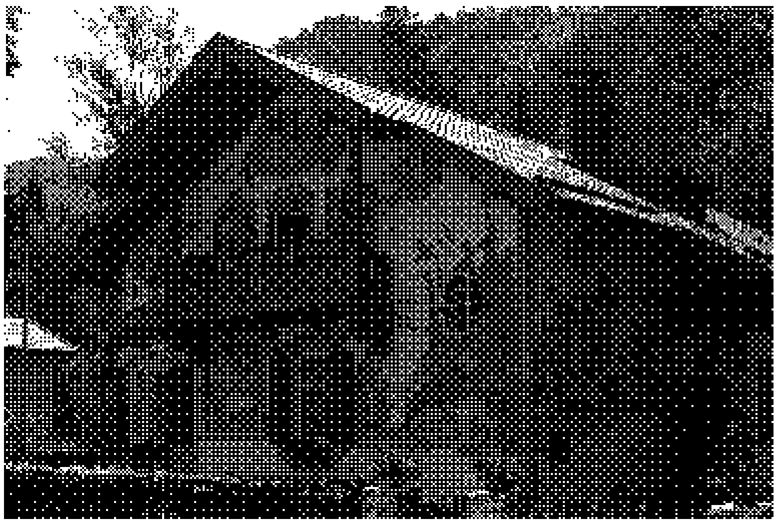


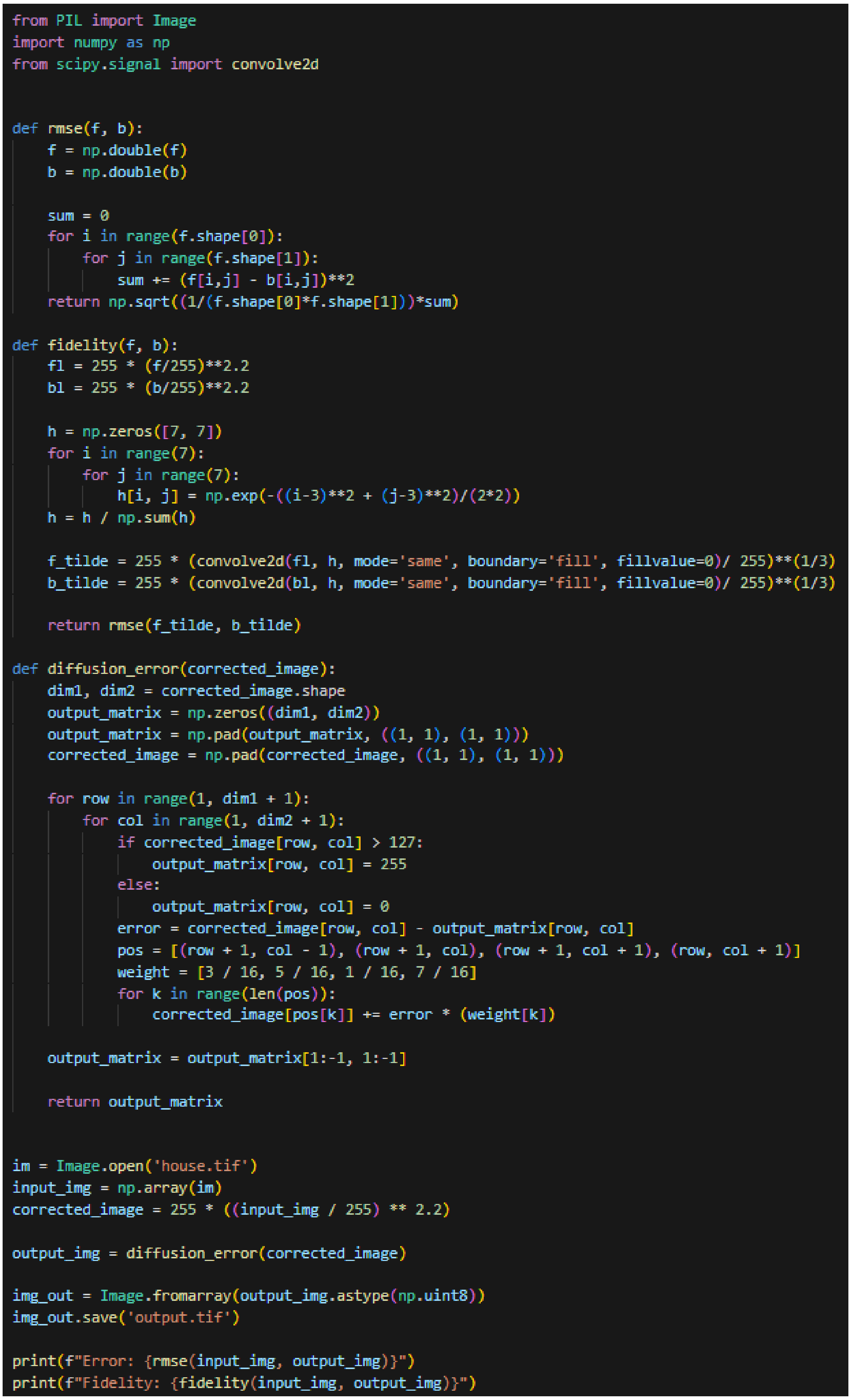
Fig 5: The halftone image produced by Bayer dithering of size 8*8

4.3. The RMSE and fidelity for each of the three halftoned images

Size of bayer dithering	RMSE	Fidelity
2*2	97.6689	50.0569
4*4	101.0069	16.5583
8*8	100.9145	14.6917

5. Error Diffusion

5.1. Your error diffusion Python code
(see next page)



```python
from PIL import Image
import numpy as np
from scipy.signal import convolve2d


def rmse(f, b):
    f = np.double(f)
    b = np.double(b)

    sum = 0
    for i in range(f.shape[0]):
        for j in range(f.shape[1]):
            sum += (f[i,j] - b[i,j])**2
    return np.sqrt((1/(f.shape[0]*f.shape[1]))*sum)

def fidelity(f, b):
    fl = 255 * (f/255)**2.2
    bl = 255 * (b/255)**2.2

    h = np.zeros([7, 7])
    for i in range(7):
        for j in range(7):
            h[i, j] = np.exp(-((i-3)**2 + (j-3)**2)/(2*2))
    h = h / np.sum(h)

    f_tilde = 255 * (convolve2d(fl, h, mode='same', boundary='fill', fillvalue=0)/ 255)**(1/3)
    b_tilde = 255 * (convolve2d(bl, h, mode='same', boundary='fill', fillvalue=0)/ 255)**(1/3)

    return rmse(f_tilde, b_tilde)

def diffusion_error(corrected_image):
    dim1, dim2 = corrected_image.shape
    output_matrix = np.zeros((dim1, dim2))
    output_matrix = np.pad(output_matrix, ((1, 1), (1, 1)))
    corrected_image = np.pad(corrected_image, ((1, 1), (1, 1)))

    for row in range(1, dim1 + 1):
        for col in range(1, dim2 + 1):
            if corrected_image[row, col] > 127:
                output_matrix[row, col] = 255
            else:
                output_matrix[row, col] = 0
            error = corrected_image[row, col] - output_matrix[row, col]
            pos = [(row + 1, col - 1), (row + 1, col), (row + 1, col + 1), (row, col + 1)]
            weight = [3 / 16, 5 / 16, 1 / 16, 7 / 16]
            for k in range(len(pos)):
                corrected_image[pos[k]] += error * (weight[k])

    output_matrix = output_matrix[1:-1, 1:-1]

    return output_matrix


im = Image.open('house.tif')
input_img = np.array(im)
corrected_image = 255 * ((input_img / 255) ** 2.2)

output_img = diffusion_error(corrected_image)

img_out = Image.fromarray(output_img.astype(np.uint8))
img_out.save('output.tif')

print(f"Error: {rmse(input_img, output_img)}")
print(f"Fidelity: {fidelity(input_img, output_img)}")
```

5.2. The error diffusion result

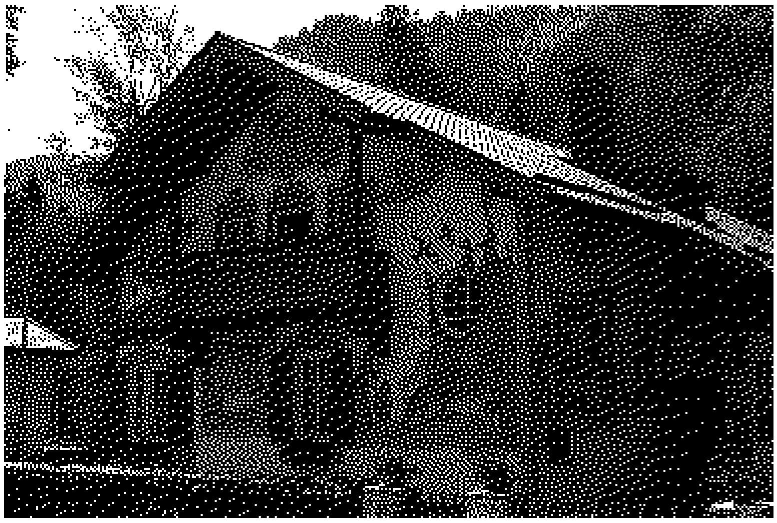


Fig 6: Error diffusion result

5.3. The RMSE and fidelity of the error diffusion result
RMSE: 98.8471
Fidelity: 13.4272

5.4. Finally, tabulate the RMSE and fidelity for the simple thresholding, ordered dithering, and error diffusion results. Comment on your observations of both the RMSE and fidelity for the different methods. Relate these metrics to the observed visual quality.

	RMSE	Fidelity
Threshold	87.3933	77.3371
2*2 Dithering	97.6689	50.0569
4*4 Dithering	101.0069	16.5583
8*8 Dithering	100.9145	14.6917
Error Diffusion	98.8471	13.4272

The Thresholding method has the lowest RMSE value, but highest fidelity. From the experiments we did, we see that the higher fidelity is, the lower the quality the image is. So RMSE value cannot represent visual quality. Fidelity is a good measurement.