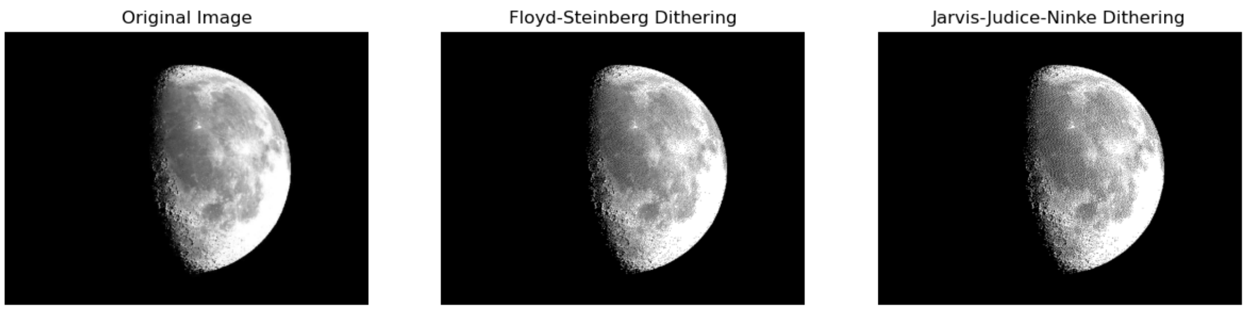
In [2]:

```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from scipy.ndimage import uniform_filter

def display_images(original, floyd, jjn):
    fig, axs = plt.subplots(1, 3, figsize=(15, 5))
    axs[0].imshow(original, cmap='gray')
    axs[0].set_title('Original Image')
    axs[0].axis('off')

    axs[1].imshow(floyd, cmap='gray')
    axs[1].set_title('Floyd-Steinberg Dithering')
    axs[1].axis('off')

    axs[2].imshow(jjn, cmap='gray')
    axs[2].set_title('Jarvis-Judice-Ninke Dithering')
    axs[2].axis('off')

    plt.show()

# Here I am using Floyd-Steinberg dithering
def floyd_steinberg_dither(img):
    img = img.copy().astype(float) / 255.0
    h, w = img.shape
    for y in range(h):
        for x in range(w):
            old_pixel = img[y, x]
            new_pixel = np.round(old_pixel)
            img[y, x] = new_pixel
            quant_error = old_pixel - new_pixel

            if x + 1 < w:
                img[y, x + 1] += quant_error * 7 / 16
            if x - 1 >= 0 and y + 1 < h:
                img[y + 1, x - 1] += quant_error * 3 / 16
            if y + 1 < h:
                img[y + 1, x] += quant_error * 5 / 16
            if x + 1 < w and y + 1 < h:
                img[y + 1, x + 1] += quant_error * 1 / 16

    return (img * 255).astype(np.uint8)

# Below I am using Jarvis-Judice-Ninke dithering
def jarvis_judice_ninke_dither(img):
    img = img.copy().astype(float) / 255.0
    h, w = img.shape
    for y in range(h):
        for x in range(w):
            old_pixel = img[y, x]
            new_pixel = np.round(old_pixel)
            img[y, x] = new_pixel
            quant_error = old_pixel - new_pixel

            diffusion_matrix = [
                    [0, 0, 0, 7/48, 5/48],
                    [3/48, 5/48, 7/48, 5/48, 3/48],
                    [1/48, 3/48, 5/48, 3/48, 1/48]
                ]

                for dy, row in enumerate(diffusion_matrix):
                    for dx, value in enumerate(row):
                        if y + dy < h and 0 <= x + (dx - 2) < w:
                            img[y + dy, x + (dx - 2)] += quant_error * value

    return (img * 255).astype(np.uint8)


image_path = '/Users/karedlashilpa/Downloads/Moon_Image.jpg'
image = Image.open(image_path).convert('L')
image_np = np.array(image)
floyd_dithered = floyd_steinberg_dither(image_np) #Now I am Applying Floy
jjn_dithered = jarvis_judice_ninke_dither(image_np) # here Applying Jarvi
display_images(image_np, floyd_dithered, jjn_dithered)
```



In [ ]:
```
#Comparision

#I compared three versions of the moon picture: the original, one using F
#I noticed that both dithering methods made the picture look smoother and
#The Floyd-Steinberg version seemed to have a more gradual transition bet
#while the Jarvis-Judice-Ninke version had a slightly different pattern o
```

In [ ]: