```
imagePath = '/MATLAB Drive/IMG_3411.JPG';
image = imread(imagePath);

figure;
imshow(image);
title('Original Image');
```

```
figure;
grayImage=rgb2gray(image);
imshow(grayImage);
title('Gray Image');
```

Floyd-Steinberg dithering algorithms

```
inputImage = double(grayImage);
% colormap
colorMap = linspace(0, 255, 16);
floydSteinbergImage = zeros(size(inputImage));
[rows, cols] = size(inputImage);

for i = 1:rows
    for j = 1:cols
        oldPixel = inputImage(i, j);
        newPixel = colorMap(find(abs(colorMap - oldPixel) ==
min(abs(colorMap - oldPixel)), 1));
        floydSteinbergImage(i, j) = newPixel;
        % Error Diffusion
        error = oldPixel - newPixel;
        % Error Distribution
        if i + 1 <= rows
             % Down pixel
            inputImage(i + 1, j) = inputImage(i + 1, j) + error * 7 / 16;
        end
        if j + 1 <= cols
            % Right  pixel
            inputImage(i, j + 1) = inputImage(i, j + 1) + error * 5 / 16;
        end
```

```
        if i - 1 >= 1 && j + 1 <= cols
            % Down-Right pixel
            inputImage(i - 1, j + 1) = inputImage(i - 1, j + 1) + error *
3 / 16;
        end
        if i + 1 <= rows && j + 1 <= cols
            % Up-Right pixel
            inputImage(i + 1, j + 1) = inputImage(i + 1, j + 1) + error *
1 / 16;
        end
    end
end
```

Jarvis-Judice-Ninke dithering algorithm

```
inputImage = double(grayImage);
colorMap = linspace(0, 255, 16);
jarvisJudiceNinkeImage = zeros(size(inputImage));

for i = 1:rows
    for j = 1:cols
        oldPixel = inputImage(i, j);
        newPixel = colorMap(find(abs(colorMap - oldPixel) ==
min(abs(colorMap - oldPixel)), 1));
        jarvisJudiceNinkeImage(i, j) = newPixel;
        error = oldPixel - newPixel;

        if i + 1 <= rows
             % Down pixel
            inputImage(i + 1, j) = inputImage(i + 1, j) + error * 7 / 48;
        end
        if i - 1 >= 1 && j + 1 <= cols
            % Right pixel
            inputImage(i - 1, j + 1) = inputImage(i - 1, j + 1) + error *
3 / 48;
        end
        if j + 1 <= cols
            % Up-Right pixel
            inputImage(i, j + 1) = inputImage(i, j + 1) + error * 5 / 48;
        end
        if i + 1 <= rows && j + 1 <= cols
            % Down-Right pixel
            inputImage(i + 1, j + 1) = inputImage(i + 1, j + 1) + error *
1 / 48;
        end
        if i - 2 >= 1 && j + 1 <= cols
            % Two rows up, right pixel
            inputImage(i - 2, j + 1) = inputImage(i - 2, j + 1) + error *
1 / 48;
        end
```

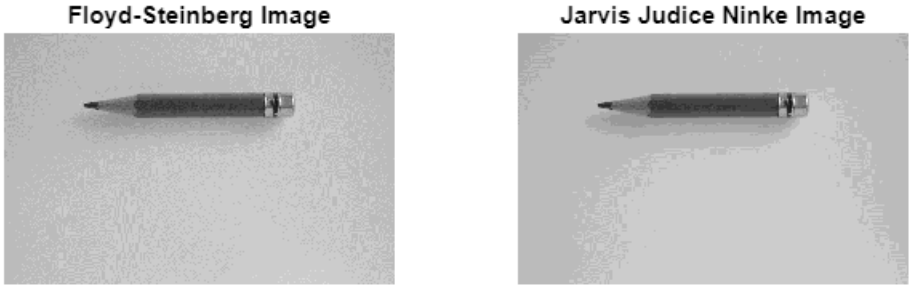
```
        if i + 2 <= rows && j + 1 <= cols
            % Two rows down, right pixel
            inputImage(i + 2, j + 1) = inputImage(i + 2, j + 1) + error *
1 / 48;
        end
    end
end
```

Result :

```
figure;

subplot(1, 2, 1);
imshow(uint8(floydSteinbergImage));
title("Floyd-Steinberg Image");

subplot(1, 2, 2);
imshow(uint8(jarvisJudiceNinkeImage));
title("Jarvis Judice Ninke Image");
```



Visual Quality Floyd-Steinberg Image Produces smoother than Jarvis Judice Ninke

Processing Time Jarvis Judice Ninke Take more Time

Floyd-Steinberg might be better for photographic images

Jarvis Judice Ninke might be better for artistic images