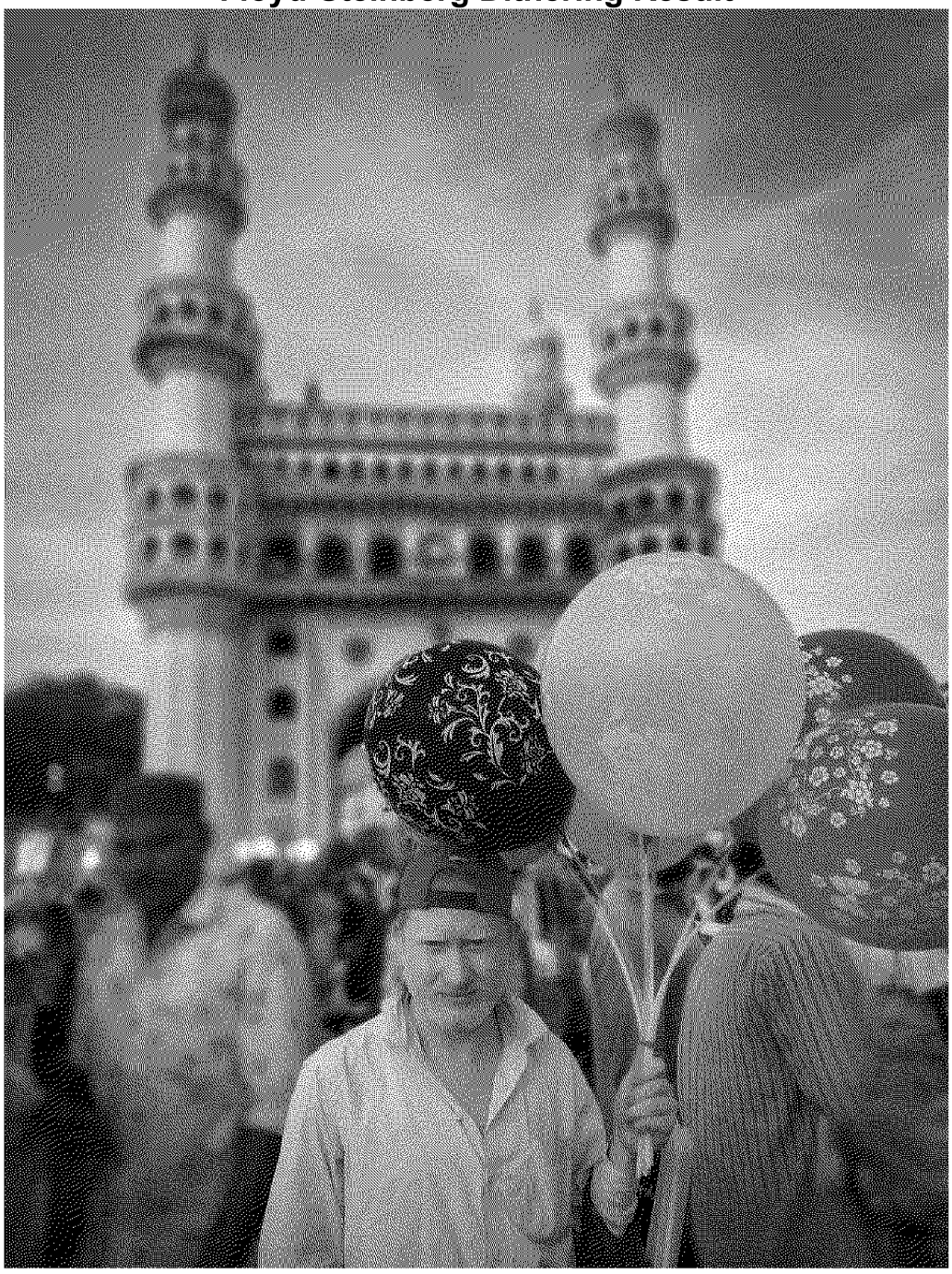
```
% Reading the image from the given path and converting it to grayscale
input_image = imread('C:\Users\USER\Pictures\BQPE6282.JPG');
grayscale_image = rgb2gray(input_image);
grayscale_image = double(grayscale_image);  % Converting to double for precision

% Function to implement Floyd-Steinberg dithering
function floyd_output = floyd_steinberg_dithering(img)
    [height, width] = size(img);
    for row = 1:height
        for col = 1:width
            current_pixel = img(row, col);
            dithered_pixel = 255 * (current_pixel > 127); % Thresholding the pixel
            img(row, col) = dithered_pixel;
            pixel_error = current_pixel - dithered_pixel;

            % Propagating the error to adjacent pixels
            if col + 1 <= width
                img(row, col + 1) = img(row, col + 1) + pixel_error * 7 / 16;
            end
            if row + 1 <= height
                if col - 1 >= 1
                    img(row + 1, col - 1) = img(row + 1, col - 1) + pixel_error *
3 / 16;
                end
                img(row + 1, col) = img(row + 1, col) + pixel_error * 5 / 16;
                if col + 1 <= width
                    img(row + 1, col + 1) = img(row + 1, col + 1) + pixel_error *
1 / 16;
                end
            end
        end
    end
    floyd_output = uint8(img);  % Return as uint8 for image display
end

% Function to implement Jarvis-Judice-Ninke dithering
function jjn_output = jarvis_judice_ninke_dithering(img)
    [height, width] = size(img);
    diffusion_kernel = [0 0 0 7 5;
                        3 5 7 5 3;
                        1 3 5 3 1] / 48;  % Normalized kernel for error diffusion

    for row = 1:height
        for col = 1:width
            current_pixel = img(row, col);
            dithered_pixel = 255 * (current_pixel > 127);  % Thresholding the pixel
            img(row, col) = dithered_pixel;
            pixel_error = current_pixel - dithered_pixel;

            % Diffusing error across the kernel
```

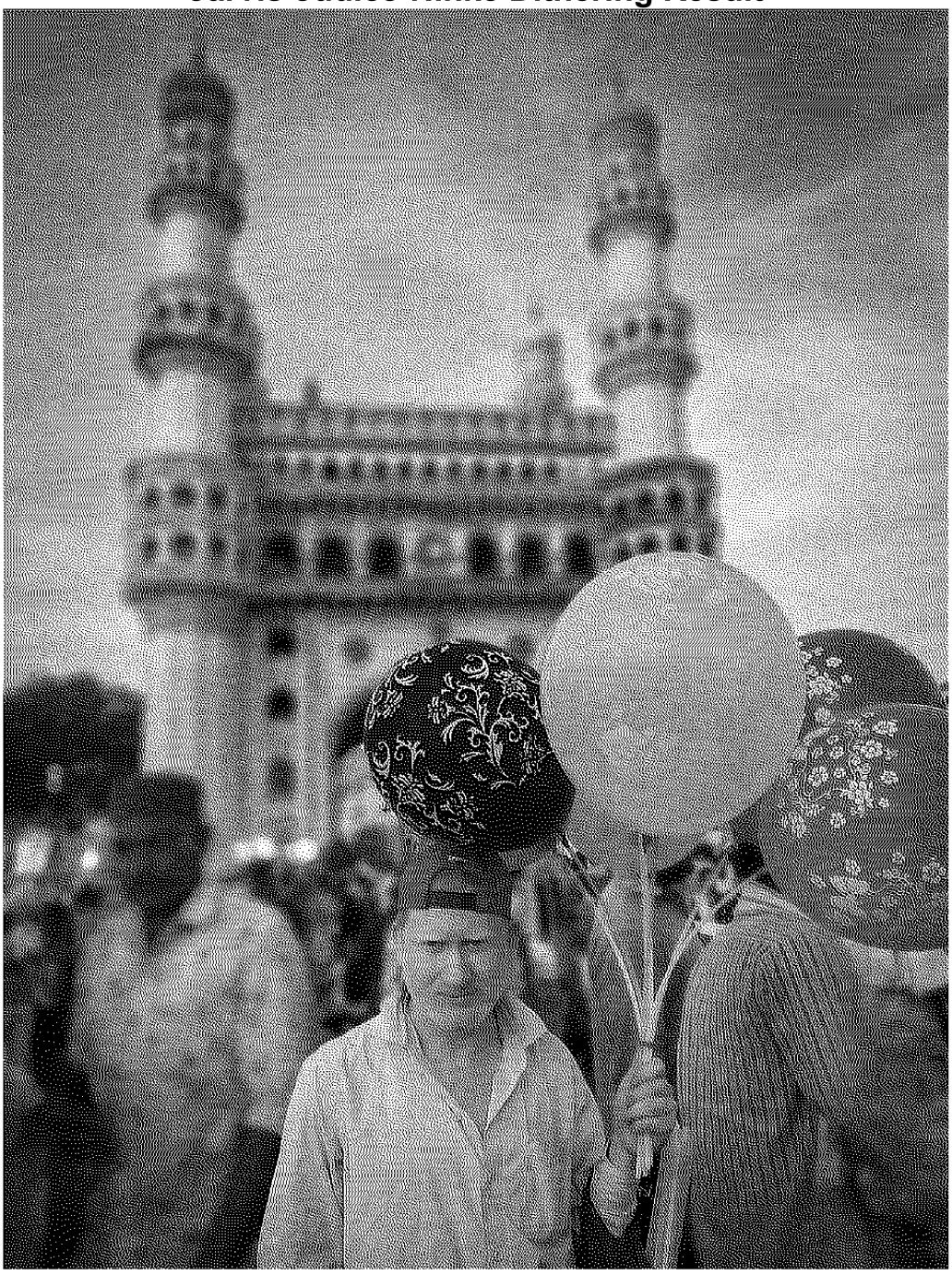
```
            for kernel_row = 1:3
                for kernel_col = 1:5
                    y = row + kernel_row - 1;
                    x = col + kernel_col - 3;
                    if y <= height && x >= 1 && x <= width
                        img(y, x) = img(y, x) + pixel_error *
diffusion_kernel(kernel_row, kernel_col);
                    end
                end
            end
        end
    end
    jjn_output = uint8(img);  % Return as uint8 for image display
end

% Applying Floyd-Steinberg Dithering and saving the result
floyd_dithered_image = floyd_steinberg_dithering(grayscale_image);
imwrite(floyd_dithered_image, 'C:\Users\USER\Desktop\floyd_dithered_output.png');
figure, imshow(floyd_dithered_image), title('Floyd-Steinberg Dithering Result');
```

Floyd-Steinberg Dithering Result



```
% Applying Jarvis-Judice-Ninke Dithering and saving the result
jjn_dithered_image = jarvis_judice_ninke_dithering(grayscale_image);
imwrite(jjn_dithered_image, 'C:\Users\USER\Desktop\jjn_dithered_output.png');
figure, imshow(jjn_dithered_image), title('Jarvis-Judice-Ninke Dithering Result');
```

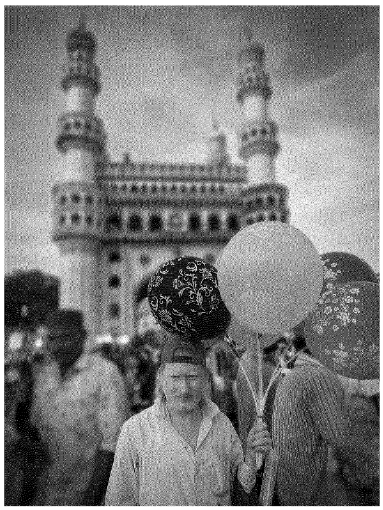
Jarvis-Judice-Ninke Dithering Result



```
% Displaying the two dithered results side by side for comparison
figure;
subplot(1, 2, 1);
imshow(floyd_dithered_image), title('Floyd-Steinberg');
subplot(1, 2, 2);
imshow(jjn_dithered_image), title('Jarvis-Judice-Ninke');
```

Floyd-Steinberg

Jarvis-Judice-Ninke



```
% Calculating and displaying the pixel-wise difference between the two results
difference_image = abs(double(floyd_dithered_image) - double(jjn_dithered_image));
figure, imshow(uint8(difference_image)), title('Difference Between Floyd-Steinberg
and JJN');
```

Difference Between Floyd-Steinberg and JJN