What is Kuwahara filter?

Kuwahara filter is a method for edge-preserving noise reduction. Calculating the mean and variance of each quadrant is how it operates. The area surrounding each pixel is divided into four quadrants. Subsequently, the filter substitutes the central pixel with the average value of the quadrant with the minimum variance. By doing so, the image's crisp edges are preserved and noise is reduced. For maintaining texture boundaries and lowering noise in regions of consistent intensity, the Kuwahara filter works especially well. Enhancing image quality without sacrificing crucial information is a common application for it in medical imaging, including MRI and ultrasound scans. Photographs can be given a comic or painterly look by applying the filter to creative effects. The Kuwahara filter has the disadvantage of occasionally producing a patchy or blocky appearance in regions with a lot of detail. The degree of smoothing effect can be varied by varying the size of the filter kernel. There are other Kuwahara filter variations, such as anisotropic ones that adjust to specific local image structures. The Kuwahara filter is useful in many image processing applications due to its ability to maintain edges, despite being computationally more costly than simple smoothing filters. It bears the name of Kuwahara Masanoba, its creator, who popularized the method in the 1970s.

```
function demonstrate_kuwahara_effect(source_image_path)
    % Read the image
    raw_input = imread(source_image_path);
    if size(raw_input, 3) == 3
        grayscale_input = rgb2gray(raw_input);
    else
        grayscale_input = raw_input;
    end
    normalized_input = im2double(grayscale_input);

    % Apply Kuwahara filter
    filter_dimension = 5;  % Size of the filter window
    kuwahara_output = apply_kuwahara_filter(normalized_input,
filter_dimension);

    % Display results
    figure;
    subplot(1,2,1), imshow(normalized_input), title('Original Image');
    subplot(1,2,2), imshow(kuwahara_output), title('Kuwahara Filtered
Result');

    % Save result
    imwrite(kuwahara_output, 'kuwahara_processed_image.png');
end

function filtered_result = apply_kuwahara_filter(source_matrix,
filter_dimension)
    [matrix_height, matrix_width] = size(source_matrix);
    filtered_result = zeros(matrix_height, matrix_width);

    radius = floor(filter_dimension / 2);

    padded_matrix = padarray(source_matrix, [radius radius], 'replicate');

    for vertical_idx = 1:matrix_height
        for horizontal_idx = 1:matrix_width
            % Extract the window
            current_window =
padded_matrix(vertical_idx:vertical_idx+filter_dimension-1,
horizontal_idx:horizontal_idx+filter_dimension-1);

            % Define the four quadrants
            upper_left_quad = current_window(1:radius+1, 1:radius+1);
            upper_right_quad = current_window(1:radius+1, radius+1:end);
            lower_left_quad = current_window(radius+1:end, 1:radius+1);
            lower_right_quad = current_window(radius+1:end, radius+1:end);

            % Calculate mean and variance for each quadrant
            quad_means = [mean(upper_left_quad(:)),
mean(upper_right_quad(:)), ...
```

```
                            mean(lower_left_quad(:)),
mean(lower_right_quad(:))];
            quad_variances = [var(upper_left_quad(:)),
var(upper_right_quad(:)), ...
                                var(lower_left_quad(:)),
var(lower_right_quad(:))];

            % Find the quadrant with the smallest variance
            [~, min_variance_index] = min(quad_variances);

            % Set the output pixel to the mean of the quadrant with the
smallest variance
            filtered_result(vertical_idx, horizontal_idx) =
quad_means(min_variance_index);
        end
    end
end

% Call the function with your image path
demonstrate_kuwahara_effect 'https://tse2.mm.bing.net/th?
id=OIP.XGOSeihZUXjgFbeRLN8xlAHaE8&pid=Api&P=0&h=180');
```

Original Image

Kuwahara Filtered Result