Question-3

3.Explain what a Kuwahara filter is, and apply it to the image using either Python or MATLAB to demonstrate its effect.

Explanation:

A Kuwahara filter is an edge-preserving smoothing filter used in image processing. It's particularly effective at reducing image noise while maintaining edge sharpness, which makes it useful for tasks like image segmentation or creating artistic effects. Here's how it works:

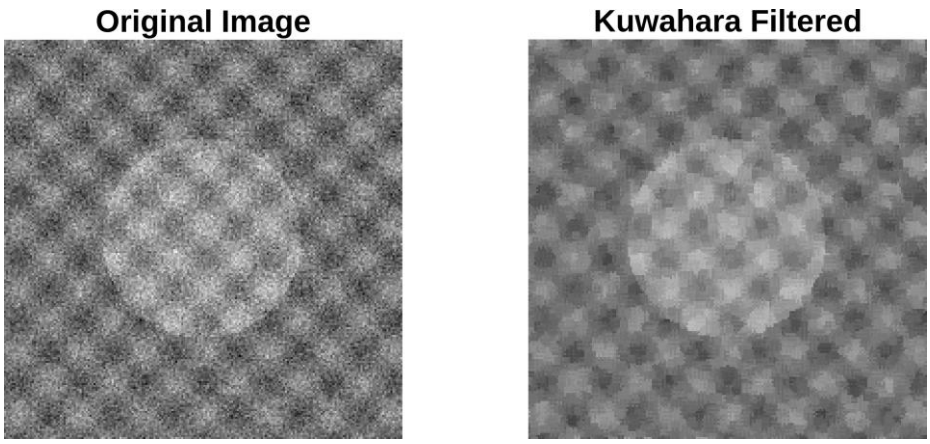
1. For each pixel in the image, the filter considers a square neighborhood around it.
2. This neighborhood is divided into four overlapping sub-regions (typically square or circular).
3. For each sub-region, the filter calculates two statistics: the mean and the variance.
4. The filter then replaces the central pixel with the mean value from the sub-region that has the smallest variance.

The key idea is that the sub-region with the smallest variance is likely to be the most homogeneous area, so using its mean helps preserve edges while smoothing out noise in flatter regions.

Now, let's implement the Kuwahara filter in MATLAB and apply it to an image to demonstrate its effect.

```
img_size = 200;
[X, Y] = meshgrid(1:img_size, 1:img_size);
center = img_size/2;
radius = img_size/4;
img = double(sqrt((X-center).^2 + (Y-center).^2) <= radius);
img = img + 0.5*randn(img_size);
img = mat2gray(img);  % Normalize to [0, 1]

% Add some texture and additional noise to make the effect more visible
[X, Y] = meshgrid(1:img_size, 1:img_size);
texture = sin(X/5) .* cos(Y/5);
img = img + 0.2 * texture + 0.1 * randn(img_size);
img = mat2gray(img);  % Normalize again to [0, 1]

% Kuwahara filter function
function output = kuwahara_filter(input, window_size)
    [h, w] = size(input);
    output = zeros(h, w);
    pad = floor(window_size/2);
    padded = padarray(input, [pad pad], 'replicate');

    for i = 1:h
        for j = 1:w
            window = padded(i:i+window_size-1, j:j+window_size-1);

            % Define the four sub-regions
            r1 = window(1:pad+1, 1:pad+1);
            r2 = window(1:pad+1, pad+1:end);
            r3 = window(pad+1:end, 1:pad+1);
            r4 = window(pad+1:end, pad+1:end);

            % Calculate mean and variance for each sub-region
            m1 = mean(r1(:)); v1 = var(r1(:));
            m2 = mean(r2(:)); v2 = var(r2(:));
            m3 = mean(r3(:)); v3 = var(r3(:));
            m4 = mean(r4(:)); v4 = var(r4(:));

            % Find the sub-region with minimum variance
            [~, idx] = min([v1, v2, v3, v4]);
            means = [m1, m2, m3, m4];

            % Set the output pixel to the mean of the chosen sub-region
            output(i, j) = means(idx);
        end
    end
end

% Apply Kuwahara filter
kuwahara_filtered = kuwahara_filter(img, 5);
```

```
% Display results
figure;
subplot(1,2,1), imshow(img), title('Original Image');
subplot(1,2,2), imshow(kuwahara_filtered), title('Kuwahara Filtered');
```




```
% Calculate and display PSNR and SSIM
psnr_value = psnr(kuwahara_filtered, img);
ssim_value = ssim(kuwahara_filtered, img);

fprintf('Peak Signal-to-Noise Ratio (PSNR): %.2f dB\n', psnr_value);
```

```
Peak Signal-to-Noise Ratio (PSNR): 21.78 dB
```

```
fprintf('Structural Similarity Index (SSIM): %.4f\n', ssim_value);
```

```
Structural Similarity Index (SSIM): 0.4372
```