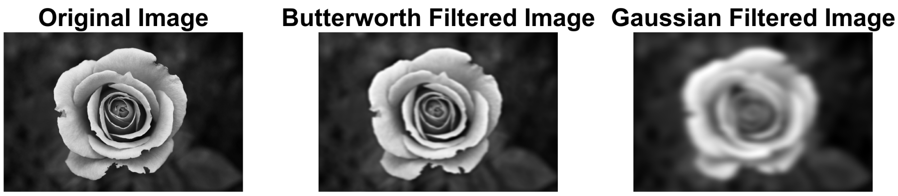
```
% Main script to load image, apply filters, and plot results
img = load_and_convert_image('input1.jpg');  % Load and convert image to grayscale

[rows, cols] = size(img);
dft_shift = compute_dft(img);  % Compute DFT and shift

% Apply Butterworth filter
cutoff = 50;
order = 2;
filtered_img_butterworth = apply_butterworth_filter_alternative(dft_shift, rows,
cols, cutoff, order);

% Apply Gaussian filter
sigma = 10;
filtered_img_gaussian = apply_gaussian_filter_alternative(dft_shift, rows, cols,
sigma);

% Plot results
plot_images(img, filtered_img_butterworth, filtered_img_gaussian);
```



```
% ---- Functions ----

% Function to load and convert the image to grayscale
function gray_img = load_and_convert_image(filename)
    img = imread(filename);
    gray_img = rgb2gray(img);  % Convert to grayscale
end

% Function to compute the DFT and shift the zero-frequency component
function dft_shift = compute_dft(img)
    dft = fft2(double(img));
    dft_shift = fftshift(dft);  % Shift zero-frequency component to center
end

% Function to apply Butterworth filter with alternative strategy
function filtered_img = apply_butterworth_filter_alternative(dft_shift, rows, cols,
cutoff, order)
    % Create frequency grid using matrix algebra
    u = ((0:cols-1) - floor(cols/2)).^2;
    v = ((0:rows-1) - floor(rows/2)).^2;

    % Create distance matrix using outer sums
    [U, V] = meshgrid(u, v);
    D2 = U + V;  % Distance squared

    % Apply Butterworth filter formula
    H = 1 ./ (1 + (sqrt(D2) / cutoff).^(2 * order));

    % Filter the DFT and inverse to get the filtered image
    filtered_dft = dft_shift .* H;
    filtered_img = real(ifft2(ifftshift(filtered_dft)));
end

% Function to apply Gaussian filter with alternative strategy
function filtered_img = apply_gaussian_filter_alternative(dft_shift, rows, cols,
sigma)
    % Create frequency grid using matrix algebra
    u = ((0:cols-1) - floor(cols/2)).^2;
    v = ((0:rows-1) - floor(rows/2)).^2;

    % Create distance matrix using outer sums
    [U, V] = meshgrid(u, v);
    D2 = U + V;  % Distance squared

    % Apply Gaussian filter formula
    H = exp(-D2 / (2 * sigma^2));

    % Filter the DFT and inverse to get the filtered image
    filtered_dft = dft_shift .* H;
    filtered_img = real(ifft2(ifftshift(filtered_dft)));
end

% Function to plot original and filtered images
function plot_images(original_img, butterworth_img, gaussian_img)
    figure;
    subplot(1, 3, 1);
    imshow(original_img, []);
    title('Original Image');

    subplot(1, 3, 2);
    imshow(butterworth_img, []);
    title('Butterworth Filtered Image');

    subplot(1, 3, 3);
    imshow(gaussian_img, []);
    title('Gaussian Filtered Image');
end
```