PARAMETERS TO ANALYZE THE EFFECTIVENESS OF FILTERS

To evaluate how well each filter removes noise while preserving image quality, use the following parameters in MATLAB:

1. **Peak Signal-to-Noise Ratio (PSNR)**:
    - **Formula**: PSNR = 10 * log10((MAX_I^2) / MSE), where MAX_I is the maximum possible pixel value (255 for 8-bit images), and MSE is the Mean Squared Error.
    - **MATLAB Function**: `psnr(denoised_image, original_image)`.
    - **Interpretation**: Higher PSNR (e.g., >30 dB) indicates better denoising with less distortion. Compare PSNR between noisy and denoised images to assess improvement.
    - **Use**: Measure how close the denoised image is to the original.

2. **Mean Squared Error (MSE)**:
    - **Formula**: MSE = (1/m) * Σ[(original(i,j) - denoised(i,j))^2], where m is the number of pixels.
    - **MATLAB Function**: `immse(denoised_image, original_image)`.
    - **Interpretation**: Lower MSE values indicate less error and better denoising. A value close to 0 is ideal.
    - **Use**: Quantify the average squared difference between original and denoised images.

3. **Structural Similarity Index (SSIM)**:
    - **Formula**: SSIM compares luminance, contrast, and structure between two images, ranging from -1 to 1 (1 being perfect similarity).

- **MATLAB Function**: Use the Image Processing Toolbox function `ssim(denoised_image, original_image)`.
- **Interpretation**: SSIM > 0.9 suggests good structural preservation; lower values indicate loss of detail or structure.
- **Use**: Assess how well the filter preserves the image's structural information.

4. **Edge Preservation (Gradient Magnitude)**:
   - **Method**: Compute the gradient magnitude of the original and denoised images using imgradient or edge functions.
   - **Interpretation**: Higher correlation (close to 1) indicates better edge preservation.
   - **Use**: Evaluate if the filter blurs edges excessively.
5. **Noise Reduction Ratio (Optional)**:
   - **Method**: Compare the variance of the noise (noisy - original) vs. residual noise (denoised - original).
   - **MATLAB Approach**:

```
noise_variance = var(double(noisy_sp(:)) - double(original(:)));

residual_variance = var(double(denoised_sp(:)) - double(original(:)));

reduction_ratio = noise_variance / residual_variance;

fprintf('Noise Reduction Ratio (Salt & Pepper): %.2f¥n', reduction_ratio);
```

   - **Interpretation**: A higher ratio indicates better noise removal.
   - **Use**: Quantify how much noise is reduced by the filter.