Lab 2: Noise Filtering

Q1:

<u>**Noise reduction**</u>:

Gaussian filter: Applying a Gaussian filter before performing edge detection can **effectively reduce the noise in the image**. The Gaussian filtering smoothing process helps to remove random noise points in the image, which may be mistaken for edges.

Direct edge detection: Direct application of edge detection algorithms such as **Sobel** operators may capture noise, as these operators are very sensitive to intensity variations.

<u>**Edge sharpness**</u>:

Edge detection after Gaussian filter: Although Gaussian filtering may **blur some edges of the image** slightly, it can also enhance the contrast of the true edges, making them more prominent and clearer after thresholding.

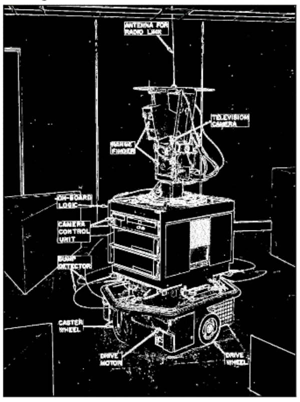
Direct edge detection: Despite being able to capture more details, directly applied edge detection may produce some **unwanted edges due to noise**.

<u>**Balance of application:**</u>

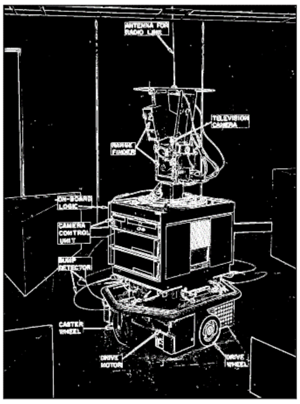
Choosing the right Gaussian kernel: Choosing the size of the Gaussian kernel is a balancing process. A larger Gaussian filter (5×5) provides more smoothing and **reduces a large part of the noise** but may **blur small edges** to some extent. On the contrary, the smaller 3×3 Gaussian filter at the same threshold (0.3) **does not reduce the noise** as much as the 5×5 Gaussian filter but **retains more edge details**.

Edge detection is performed after the application of Gaussian filters (3×3 and 5×5), and at the same time the noise is reduced at the same threshold (0.3), the edges become more pronounced compared to the direct edge detection.

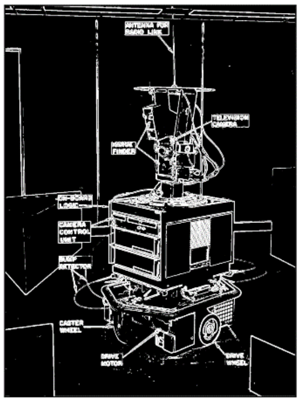
Original Sobel Thresholded 0.3



3x3 Gaussian Sobel Thresholded 0.3



5x5 Gaussian Sobel Thresholded 0.3



Q2:

<u>**Increase the size of the Gaussian filter (3×3 vs 5×5) :**</u>

Smoothing effect: As the Gaussian filter size increases, the smoothing effect of the image will be more significant. This means that a **larger filter mixes pixel values over a wider range**, resulting in **a stronger blurring effect**.

Edge detection: Using a **5×5** Gaussian filter in this example rather than a **3×3** Gaussian filter makes the noise become less salient to the true edges in edge detection

<u>**Changing the standard deviation(Std:1 vs Std:2):**</u>

Width of the Gaussian distribution: The standard deviation determines the width of the weight distribution in the Gaussian filter. A **larger standard deviation** results in a **wider Gaussian distribution**, implying a wider smoothing.

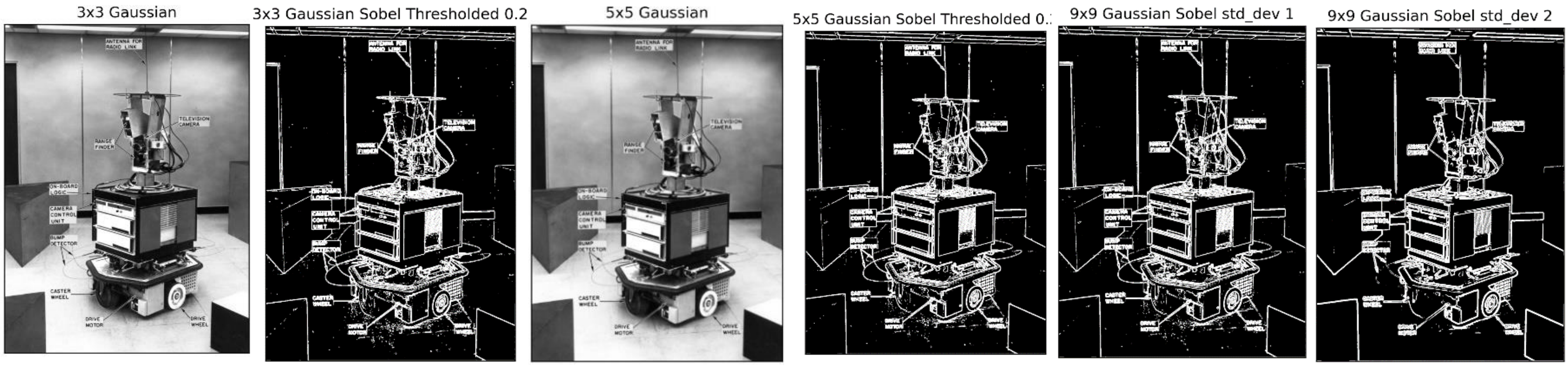
```python
def generate_2d_gaussian_filter_using_arange(size, std_dev):
    x = size // 2
    vec = np.arange(-x, x+1, step=1, dtype=np.float32)
    gaussian_1d = sample_gaussian(std_dev, 0, vec)
    gaussian_1d /= gaussian_1d.sum()
    gaussian_filter = np.outer(gaussian_1d, gaussian_1d)
    return gaussian_filter
```

Impact on edges: In this example when I use **Std=2** the noise is reduced relative to **Std=1** and the edge detection is more prominent. However, a **larger standard deviation** may also lead to more **blurred edges**, while a **smaller standard deviation** can preserve **more details** but may also retain **more noise**. Using **different thresholds** will also have different collocations with **different standard deviations** will also have **different effects.**

<u>Effect of noise filtering on edge detection:</u>

Noise reduction: Gaussian filters can effectively reduce noise in an image. Therefore, applying Gaussian filtering before edge detection can **reduce the possibility of mis-detected edges**.

Visibility of edges: Edge detection after Gaussian filtering usually results in cleaner and more consistent edges. This may be **more visible** visually, especially in more homogeneous areas.



Q3:

<u>Laplacian-only operator</u>

```python
def apply_laplacian_filter(image):
    laplacian_filter = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]])
    laplacian = scipy.signal.convolve2d(image, laplacian_filter, mode='same', boundary='wrap')
    return laplacian
```

Edge detection: The Laplacian is an edge detection method based on the second derivative, which is **very sensitive to fast intensity changes** in the image and can effectively **highlight edges**.

Noise Sensitivity: The Laplacian is very **sensitive to noise**. In noisy images, the Laplacian may misidentify noise as an edge, leading to many **false positive edges**.

Edge properties: Since the Laplacian is very sensitive to intensity changes in the image, it is able to **produce very fine and sharp edges and may also misidentify noise as edges**.

```python
def apply_laplacian_of_gaussian_filter(image, size, std_dev):
    gaussian_filter = generate_2d_gaussian_filter_using_arange(size, std_dev)
    gaussian_blur = scipy.signal.convolve2d(image, gaussian_filter, mode='same', boundary='wrap')
    laplacian = apply_laplacian_filter(gaussian_blur)
    edgedetect = zero_cross(laplacian)
    return edgedetect
```

<u>Combining Gaussian smoothing and Laplacian (LoG)</u>

Noise suppression: Gaussian smoothing before applying the Laplacian can effectively **reduce the noise** in the image. This preprocessing step helps to **reduce false detections due to noise**.

Edge smoothing: Due to the smoothing effect of the Gaussian filter, the LoG method may not scapture sharp edges like the pure Laplacian. The edges may be **slightly blurred**, but the **real edges are better displayed in noisy images**.

Laplacian-only operator: suitable for cases where edge sharpness is critical and the image is less noisy. However, it may **lead to more false detections in the case of more noise.**

Combining Gaussian smoothing with Laplacian (LoG) : Suitable for noisy images where it is necessary to **balance the accuracy of noise suppression and edge detection.** It provides a **more robust approach to edge detection, especially in noisy environments**.