# ROBOT PERCEPTION LAB 3 – FILTERS

# TASK 1


## NAME: ANIRUDDHAN

REG: 21BRS1682


**Aim:** To apply filters like gaussian, box, median and mean filter for image smoothening/Enhancement and reducing noise.

**Algorithm:**

**Step 1: Import Required Libraries**

1. Import necessary libraries:
    - numpy: For handling arrays and numerical operations.
    - scipy.ndimage: For applying various filters (uniform, median, Gaussian).
    - skimage.io: For reading and saving images.
    - skimage.color: For converting color images to grayscale.
    - skimage.util: For utility functions like converting images to 8-bit unsigned integers.
    - matplotlib.pyplot: For displaying images.

**Step 2: Load and Convert Image to Grayscale**

1. Load the image from a specified file path using io.imread().
2. Convert the loaded image to grayscale using color.rgb2gray(). This conversion simplifies the image, reducing it to a single channel, which is easier to process with filters.

**Step 3: Define Filtering Functions**

1. **Mean Filter Function:**
    - Define a function mean_filter(image, size=5) that takes an image and filter size as input.
    - Use ndi.uniform_filter() to apply a mean filter, which smooths the image by averaging pixel values within a defined window.
2. **Median Filter Function:**
    - Define a function median_filter(image, size=5) that takes an image and filter size as input.
    - Use ndi.median_filter() to apply a median filter, which replaces each pixel's value with the median of the neighboring pixels, reducing noise.
3. **Gaussian Filter Function:**
    - Define a function gaussian_filter(image, sigma=2.0) that takes an image and sigma value as input.
    - Use ndi.gaussian_filter() to apply a Gaussian filter, which smooths the image by applying a Gaussian function, giving more weight to nearby pixels.
4. **Box Filter Function:**
    - Define a function box_filter(image, size=7) that takes an image and filter size as input.
    - Use uniform_filter() to apply a box filter, similar to the mean filter but optimized for larger windows.

**Step 4: Apply Filters to the Grayscale Image**

1. Apply each filter function to the grayscale image:

- Call mean_filter(gray_image) and store the result in mean_result.
- Call median_filter(gray_image) and store the result in median_result.
- Call gaussian_filter(gray_image) and store the result in gaussian_result.
- Call box_filter(gray_image) and store the result in box_result.

**Step 5: Save Filtered Images**

1. Convert the filtered images to 8-bit unsigned integers using util.img_as_ubyte() for compatibility with standard image formats.
2. Save each filtered image to disk using io.imsave():
    - Save mean_result as "mean_filtered_image.jpg".
    - Save median_result as "median_filtered_image.jpg".
    - Save gaussian_result as "gaussian_filtered_image.jpg".
    - Save box_result as "box_filtered_image.jpg".

**Step 6: Display Filtered Images**

1. Define a function display_image(image, title) that takes an image and title as input.
    - Use plt.figure() to create a new figure for each image.
    - Display the image using plt.imshow() with the grayscale colormap (cmap='gray').
    - Set the title using plt.title() and hide the axis using plt.axis('off').
    - Use plt.show() to display the image.

2. Call display_image() for each filtered image to visualize the results:
    - Display mean_result with the title "Mean Filter".
    - Display median_result with the title "Median Filter".
    - Display gaussian_result with the title "Gaussian Filter".
    - Display box_result with the title "Box Filter".

Code:

## 1.a Default Parameters

```
import numpy as np
import scipy.ndimage as ndi
from skimage import io, color, filters
from skimage import util
from scipy.ndimage import uniform_filter
import matplotlib.pyplot as plt

# Load and convert image to grayscale
image = io.imread("/content/img1.jpg")
gray_image = color.rgb2gray(image) # Convert to grayscale

# Mean Filter
def mean_filter(image, size=5):
    return ndi.uniform_filter(image, size=size) # Indent this line
```

```
# Median Filter
def median_filter(image, size=5):
    return ndi.median_filter(image, size=size) # Indent this line

# Gaussian Filter
def gaussian_filter(image, sigma=2.0):
    return ndi.gaussian_filter(image, sigma=sigma) # Indent this line

def box_filter(image, size=7):
    return uniform_filter(image, size=size) # Indent this line

# Apply filters
mean_result = mean_filter(gray_image)
median_result = median_filter(gray_image)
gaussian_result = gaussian_filter(gray_image)
box_result = box_filter(gray_image)

# Save or display results as needed
io.imsave('mean_filtered_image.jpg', util.img_as_ubyte(mean_result))
io.imsave('median_filtered_image.jpg', util.img_as_ubyte(median_result))
io.imsave('gaussian_filtered_image.jpg', util.img_as_ubyte(gaussian_result))
io.imsave('box_filtered_image.jpg', util.img_as_ubyte(box_result))

def display_image(image, title):
    plt.figure(figsize=(8, 8))
    plt.imshow(image, cmap='gray')
    plt.title(title)
    plt.axis('off') # Hide axis
    plt.show()
```

```
# Display results
display_image(mean_result, 'Mean Filter')
display_image(median_result, 'Median Filter')
display_image(gaussian_result, 'Gaussian Filter')
display_image(box_result, 'box Filter')
```

## O/p 1.a:

## Mean filter:

## Median Filter:

Gaussian Filter:

Box Filter:

# 1.b Change in parameters:

## Code:

```
import numpy as np
import scipy.ndimage as ndi
from skimage import io, color, filters
from skimage import util
from scipy.ndimage import uniform_filter
import matplotlib.pyplot as plt

# Load and convert image to grayscale
image = io.imread("/content/img1.jpg")
gray_image = color.rgb2gray(image) # Convert to grayscale

# Mean Filter
def mean_filter(image, size=1):
    return ndi.uniform_filter(image, size=size) # Indent this line

# Median Filter
def median_filter(image, size=3):
    return ndi.median_filter(image, size=size) # Indent this line

# Gaussian Filter
def gaussian_filter(image, sigma=1.0):
    return ndi.gaussian_filter(image, sigma=sigma) # Indent this line

def box_filter(image, size=3):
    return uniform_filter(image, size=size) # Indent this line
```

```
# Apply filters
mean_result = mean_filter(gray_image)
median_result = median_filter(gray_image)
gaussian_result = gaussian_filter(gray_image)
box_result = box_filter(gray_image)

# Save or display results as needed
io.imsave('mean_filtered_image.jpg', util.img_as_ubyte(mean_result))
io.imsave('median_filtered_image.jpg', util.img_as_ubyte(median_result))
io.imsave('gaussian_filtered_image.jpg', util.img_as_ubyte(gaussian_result))
io.imsave('box_filtered_image.jpg', util.img_as_ubyte(box_result))

def display_image(image, title):
    plt.figure(figsize=(8, 8))
    plt.imshow(image, cmap='gray')
    plt.title(title)
    plt.axis('off') # Hide axis
    plt.show()

# Display results
display_image(mean_result, 'Mean Filter')
display_image(median_result, 'Median Filter')
display_image(gaussian_result, 'Gaussian Filter')
display_image(box_result, 'box Filter')
```

O/P 1.b:

Mean Filter:


Median Filter:

Gaussian Filter:

Box Filter: