Experiment 4

Geometric Transformation and Filtering Using OpenCV

Aim:

To implement the following Geometric Transformations a Filtering functions on an image in Open CV:

a) Translation, Rotation, Affine Transformation and Perspective Transformation
b) 2D convolution, Averaging and Blurring
c) Thresholding

Software/ Packages Used:

1. Pycharm IDE
2. Libraries used:
    - NumPy
    - opencv-python
    - matplotlib
    - scipy

Programs:

1)Geometric Transformations

#Translation
#Rotation
Affine Transformation
#Perspective Transformation

2)Filtering

#Image Blurring
#Gausian Blurring
#Median Blurring
#Bilateral Filtering
#Average Blurring

3)Thresholding

#Simple
#Adaptive
#Ostu's

1)Geometric Transformation (Output):

(Input Image- Colour)

(Output Images- Colour)

1)Geometric Transformation (Code):

```
import cv2
import numpy as np

def geometric_transformations(image_path):
    img = cv2.imread(image_path)
    rows, cols = img.shape[:2]

    # Translation
    translation_matrix = np.float32([[1, 0, 50], [0, 1, 30]]) # Shift by (50, 30)
    translated_img = cv2.warpAffine(img, translation_matrix, (cols, rows))

    # Rotation
    rotation_matrix = cv2.getRotationMatrix2D((cols / 2, rows / 2), 45, 1) # Rotateby
45 degrees
    rotated_img = cv2.warpAffine(img, rotation_matrix, (cols, rows))

    # Affine Transformation (combination of translation, rotation, scaling, and
shearing)
    affine_matrix = np.float32([[0.5, 0.5, 50], [-0.5, 0.5, 30]]) # Custom affine matrix
    affine_transformed_img = cv2.warpAffine(img, affine_matrix, (cols, rows))

    # Perspective Transformation
    perspective_matrix = np.float32([[0.5, 0.5, 50], [-0.5, 0.5, 30], [0, 0, 1]])
    perspective_transformed_img = cv2.warpPerspective(img,
perspective_matrix,(cols, rows))

    cv2.imshow('Original Image', img)
    cv2.imshow('Translated Image', translated_img)
    cv2.imshow('Rotated Image', rotated_img)
    cv2.imshow('Affine Transformed Image', affine_transformed_img)
    cv2.imshow('Perspective Transformed Image', perspective_transformed_img)

    cv2.waitKey(0)
    cv2.destroyAllWindows()

#Usage
geometric_transformations(r"C:\Users\RAGHUL\Downloads\krishna.jpg")
```

2.1) Image Blurring (Output):

(Input Image-Colour)

(Output image-Colour)

2.2) Gausian Blurring (Output):

(Input Image-Colour)

(Output image-Colour)

2.1) Image Blurring (Code):

```
import cv2
image = cv2.imread(r"C:\Users\RAGHUL\Downloads\krishna.jpg")
average_blurred_image = cv2.blur(image, (5, 5))
cv2.imshow("Original image",image )
cv2.imshow("Average Blurred Image", average_blurred_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2.2) Gausian Blurring (Code):

```
import cv2
image = cv2.imread(r"C:\Users\RAGHUL\Downloads\krishna.jpg")
blurred_image = cv2.GaussianBlur(image, (5,5), 0)
cv2.imshow("Original image",image )
cv2.imshow("Gaussian Blurred Image", blurred_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2.3) Bilateral Blurring (Output):

(Input Image-Colour) (Output image-Colour)

2.4) Median Blurring (Output):

(Input Image-Colour) (Output image-Colour)

2.3) Bilateral Blurring (Code):

```
import cv2
image = cv2.imread(r"C:\Users\RAGHUL\Downloads\krishna.jpg")
bilateral_filtered_image = cv2.bilateralFilter(image, 9, 175, 715)
cv2.imshow("Original image",image )
cv2.imshow("Bilateral Filtered Image", bilateral_filtered_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2.4) Median Blurring (Code):

```
import cv2

image = cv2.imread (r"C:\Users\RAGHUL\Downloads\krishna.jpg")
median_blurred_image = cv2.medianBlur(image, 25)
cv2.imshow("Original image",image )
cv2.imshow("Median Blurred Image", median_blurred_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2.5) User Defined Average Blurring (Output):

```
Input Matrix:
[[1 1 1]
 [1 0 1]
 [1 1 1]]

Blurred Result:
[[0.75       0.83333333 0.75      ]
 [0.83333333 0.88888889 0.83333333]
 [0.75       0.83333333 0.75      ]]
```

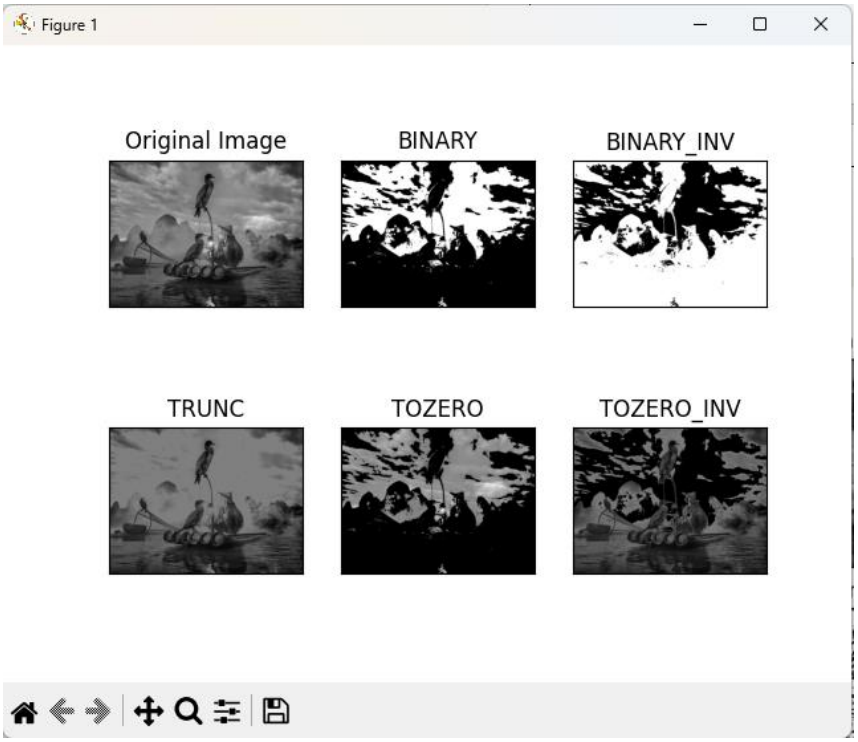
(Output in Terminal)

2.5) User Defined Average Blurring (Code):

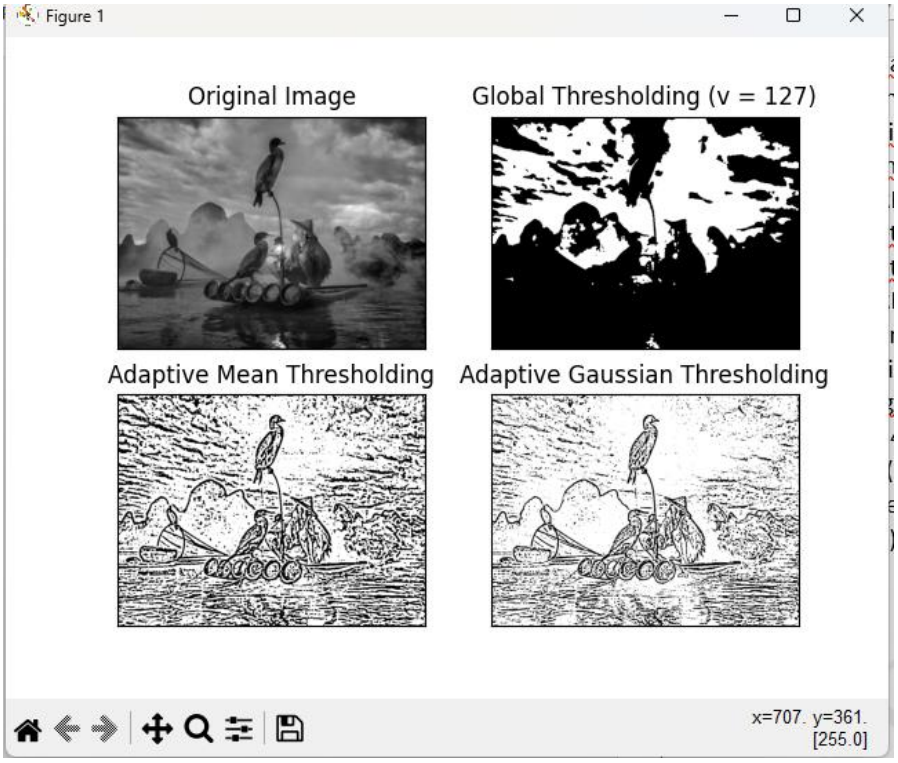
```
import numpy as np

def average_blurring(input_matrix, kernel_size):
    rows, cols = input_matrix.shape
    blurred_matrix = np.zeros_like(input_matrix, dtype=float)
    kernel_radius = kernel_size // 2
    for i in range(rows):
        for j in range(cols):
            pixel_sum = 0.0
            count = 0
            for m in range(-kernel_radius, kernel_radius + 1):
                for n in range(-kernel_radius, kernel_radius + 1):
                    if 0 <= i + m < rows and 0 <= j + n < cols:
                        pixel_sum += input_matrix[i + m, j + n]
                        count += 1
            blurred_matrix[i, j] = pixel_sum / count
    return blurred_matrix

input_matrix = np.array([[1, 1, 1],
                         [1, 0, 1],
                         [1, 1, 1]])
kernel_size = 3
blurred_result = average_blurring(input_matrix, kernel_size)

print("Input Matrix:")
print(input_matrix)
print("\nBlurred Result:")
print(blurred_result)
```

3)Thresholding (Output):

(SIMPLE THRESHOLDING)



(ADAPTIVE THRESHOLDING)



3)Thresholding (Code):

SIMPLE THRESHOLDING

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread(r"X:\BIRD.jpg", cv.IMREAD_GRAYSCALE)
assert img is not None, "file could not be read, check with os.path.exists()"
ret,thresh1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
ret,thresh2 = cv.threshold(img,127,255,cv.THRESH_BINARY_INV)
ret,thresh3 = cv.threshold(img,127,255,cv.THRESH_TRUNC)
ret,thresh4 = cv.threshold(img,127,255,cv.THRESH_TOZERO)
ret,thresh5 = cv.threshold(img,127,255,cv.THRESH_TOZERO_INV)
titles = ['Original
Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]
for i in range(6):
   plt.subplot(2,3,i+1),plt.imshow(images[i],'gray',vmin=0,vmax=255)
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
 plt.show()
```

ADAPTIVE THRESHOLDING

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread(r"X:\BIRD.jpg", cv.IMREAD_GRAYSCALE)
assert img is not None, "file could not be read, check with os.path.exists()"
img = cv.medianBlur(img,5)
ret,th1 = cv.threshold(img,127,255,cv.THRESH_BINARY),\
        cv.THRESH_BINARY,11,2)
th2 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_MEAN_C
th3 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_GAUSSIAN_C,\
        cv.THRESH_BINARY,11,2)
titles = ['Original Image', 'Global Thresholding (v = 127)',
        'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
images = [img, th1, th2, th3]
for i in range(4):
   plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
   plt.title(titles[i])
   plt.xticks([]),plt.yticks([])
plt.show()
```

(Otsu's Thresholding)

Input Image

Output Image

Otsu's Thresholding:

```
import cv2
import numpy as np

image1 = cv2.imread(r"C:\Users\RAGHUL\Downloads\krishna.jpg")

img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)

ret, thresh1 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY +
                    cv2.THRESH_OTSU)


cv2.imshow('Otsu Threshold', thresh1)

if cv2.waitKey(0) & 0xff == 27:
   cv2.destroyAllWindows()
```

Department of RAE			
Criteria	**Excellent (75% - 100%)**	**Good (50 – 75%)**	**Poor (<50%)**
Preparation (30)			
Performance (30)			
Evaluation (20)			
Report (20)			
Sign:		**Total (100)**	

Result:

Thus, the Geometrical Transformations and Filtering Techniques were learnt using OpenCV.

Post Lab Questions:

1. What is the difference between affine transformation and perspective transformation?
2. What do you mean by cartooning?
3. What does the filter2D function do? Explain with the arguments
4. Write a program for Ostu's thresholding without using inbuilt function

Post Lab Answers:

1. Difference between affine transformation and perspective transformation:

- **Affine Transformation:** Preserves parallel lines in an image. It includes operations like translation, rotation, scaling, and shearing, where lines remain parallel before and after transformation.
- **Perspective Transformation:** Involves transformations that allow perspective distortion, where parallel lines might converge or diverge. This transformation is more general and includes changes that aren't solely based on affine operations.

2. **– Cartooning** refers to the process of rendering images in a simplified or exaggerated manner, often resembling cartoons or illustrations. It involves reducing detail, enhancing outlines, and simplifying colors or shading to create a stylized or artistic effect.

3. **Explanation of the `filter2D` function with its arguments:**
- `filter2D` is an OpenCV function used for applying a custom convolution kernel to an image.
- **Arguments:**
- `src`: The input image.
- `ddepth`: Depth of the output image; set to -1 to match `src` depth.
- `kernel`: The convolution kernel, usually a numpy array.
- `anchor`: The anchor point of the kernel (default is (-1, -1), which means the center of the kernel).
- `delta`: Optional value added to the filtered pixels.
- `borderType`: Specifies the pixel extrapolation method.

4. **Program for Otsu's thresholding without using inbuilt function:**

- Here's an implementation of Otsu's thresholding without using the inbuilt `cv2.threshold` function in Python:

```
import numpy as np
import cv2

def otsu_threshold(image):
    pixel_counts = [np.sum(image == i) for i in range(256)]
    total_pixels = sum(pixel_counts)
    sum_b = 0
    weight_b = 0
    maximum = 0
    threshold = 0

    for i in range(256):
        weight_b += pixel_counts[i]
        if weight_b == 0:
            continue

        weight_f = total_pixels - weight_b
        if weight_f == 0:
            break

        sum_b += i * pixel_counts[i]
        mean_b = sum_b / weight_b
        mean_f = (sum - sum_b) / weight_f

        between = weight_b * weight_f * (mean_b - mean_f) ** 2

        if between > maximum:
            maximum = between
            threshold = i

    return threshold

# Example usage
image = cv2.imread('path/to/your/image.jpg', cv2.IMREAD_GRAYSCALE)
threshold_value = otsu_threshold(image)
ret, thresholded_image = cv2.threshold(image, threshold_value, 255, cv2.THRESH_BINARY)
cv2.imshow('Thresholded Image', thresholded_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```