COMPUTER VISION
ASSIGNMENT-2

AMOGH GARG – 2020UCO1688

IMPORTING LIBRARIES

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread("Img-1.jpg",cv2.IMREAD_COLOR)
img = cv2.resize(img, (500,500))
cv2.imshow('Colored Image of Butterfly', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Task - 1 : Perform linear filtering on an image.

```
img_blur = cv2.blur(src=img, ksize=(5,5)) # Using the blur function to blur an image
where ksize is the kernel size

# Display using cv2.imshow()
cv2.imshow('Original', img)
cv2.imshow('Blurred', img_blur)

cv2.waitKey()
cv2.imwrite('Img_Blur-1.jpg', img_blur)
cv2.destroyAllWindows()
```

Task - 2 : Perform Gaussian Filtering on an image.

```
# sigmaX is Gaussian Kernel standard deviation
# ksize is kernel size
gaussian_blur = cv2.GaussianBlur(src=img, ksize=(5,5),sigmaX=0, sigmaY=0)

cv2.imshow('Original', img)
cv2.imshow('Gaussian Blurred', gaussian_blur)

cv2.waitKey()
cv2.imwrite('Img_gaussian_blur.jpg', gaussian_blur)
cv2.destroyAllWindows()
```

Task - 3 : Compare the result of linear filter and gaussian filter by extracting a small patch from the output images.

```
from sklearn.feature_extraction import image
patches_blur = image.extract_patches_2d(img_blur, (5,5))
patches_gauss = image.extract_patches_2d(gaussian_blur, (5,5))
```

```
cv2.imshow('Linear Patch', patches_blur[50])
cv2.imshow('Gaussian Patch', patches_gauss[50])
cv2.waitKey()
cv2.destroyAllWindows()
# (Number of Patches, Height, Width, Number of Channels)
#print('Patches shape: {}'.format(patches.shape))
```

Task - 4 : Perform scaling, rotation and translation on an image.

```
scale_img = cv2.resize(img,None,fx=0.5, fy=0.5, interpolation = cv2.INTER_CUBIC)
cv2.imshow('Scaled Image', scale_img)
cv2.waitKey()
cv2.destroyAllWindows()
```

```
num_rows, num_cols = img.shape[:2]
translation_matrix = np.float32([ [1,0,70], [0,1,110] ])
trans_img = cv2.warpAffine(img, translation_matrix, (num_cols, num_rows),
cv2.INTER_LINEAR)
cv2.imshow('Translated Image', trans_img)
cv2.waitKey()
cv2.destroyAllWindows()
```

```
rotated_img = cv2.warpAffine(img, cv2.getRotationMatrix2D((num_cols/2, num_rows/2), 30,
0.6), (num_cols, num_rows))
cv2.imshow('Rotated Image', rotated_img)
cv2.waitKey()
cv2.destroyAllWindows()
```

Task - 5 : Perform Affine transformation on an image.

```
src_points = np.float32([[0,0], [num_cols-1,0], [0,num_rows-1]])
dst_points = np.float32([[0,0], [int(0.6*(num_cols-1)),0], [int(0.4*(num_cols-
1)),num_rows-1]])
matrix = cv2.getAffineTransform(src_points, dst_points)
img_afftran = cv2.warpAffine(img, matrix, (num_cols,num_rows))
cv2.imshow('Affine Transformation',img_afftran)
cv2.waitKey()
cv2.destroyAllWindows()
```

Task - 6 : Apply median filter on an image.

```
median_img = cv2.medianBlur(src=img, ksize=5)
cv2.imshow('Median Filter', median_img)
cv2.waitKey()
cv2.destroyAllWindows()
```

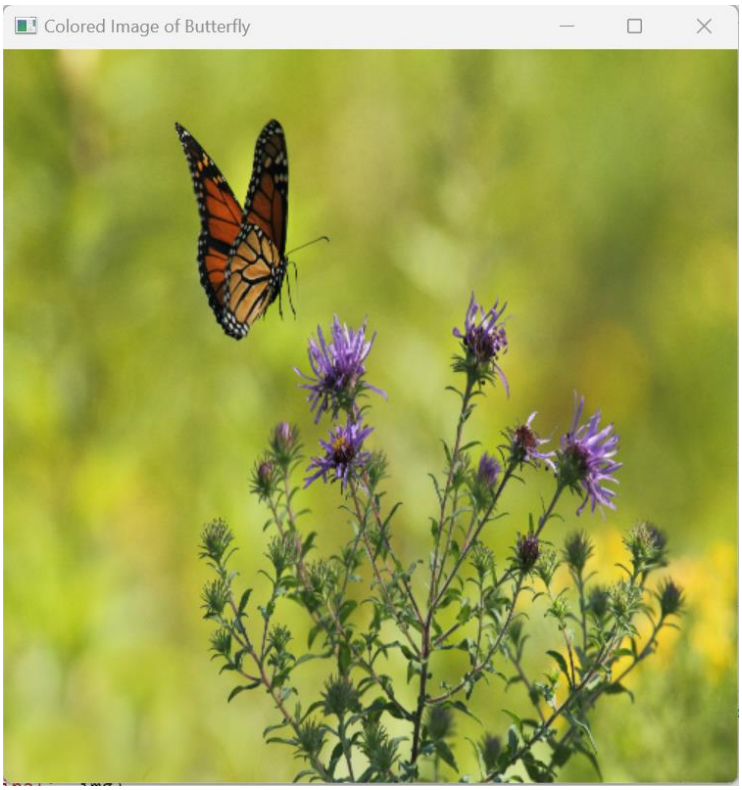
Task - 7 : What will be the impact of increasing the size of filter on the output image?

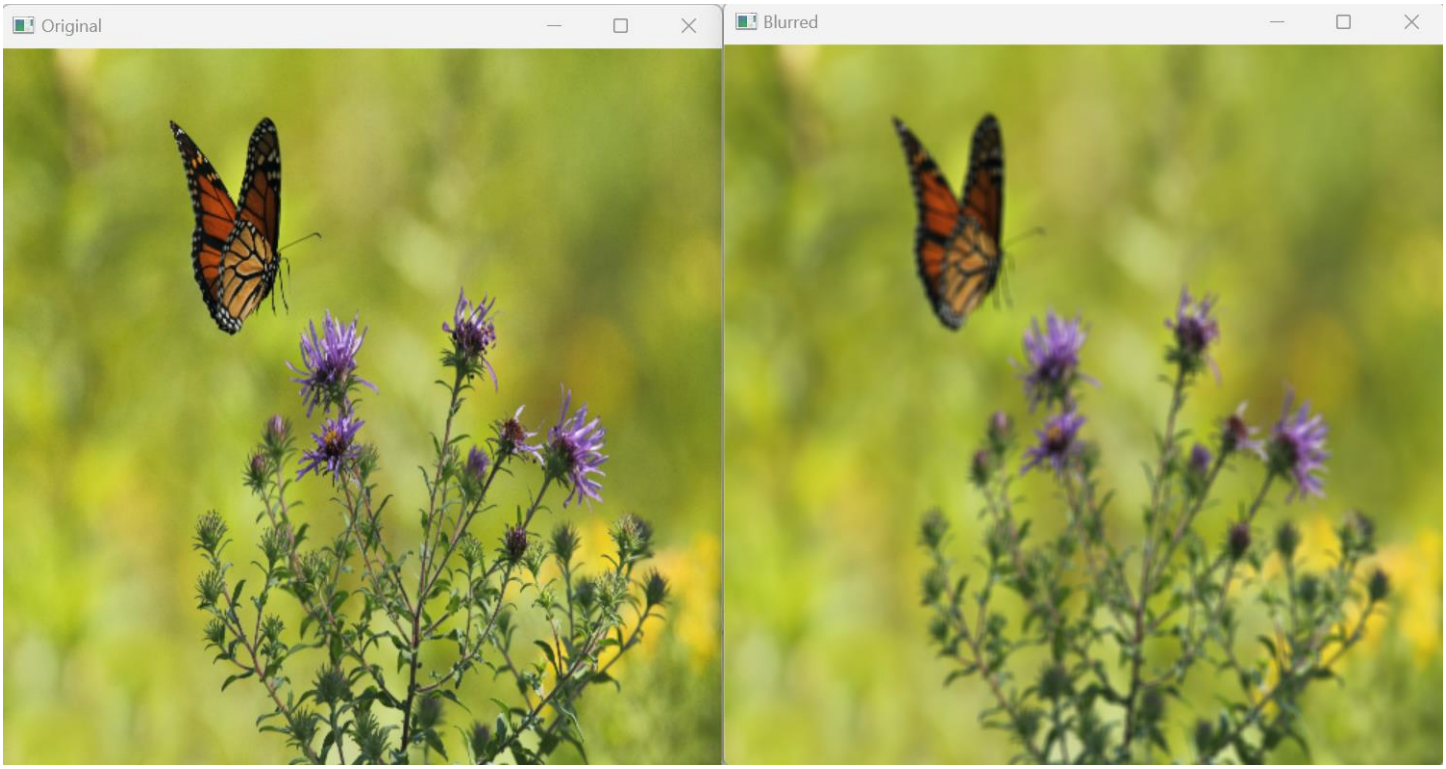
Increasing the size of a filter in image processing refers to increasing the dimensions of the kernel used in a convolution operation. This has the effect of expanding the scope of the filter, allowing it to cover more pixels in the image and therefore extract more information.

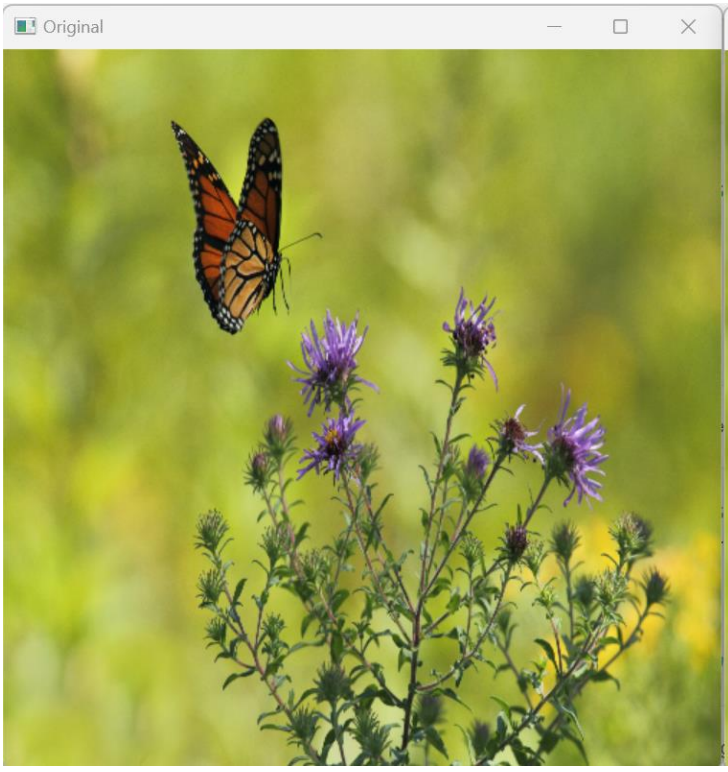
However, increasing the size of the filter can also result in loss of resolution in the output image. This is because the larger filter will blur the image more and cause information to be lost. Additionally, increasing the size of the filter can also slow down the processing time and increase computational complexity.

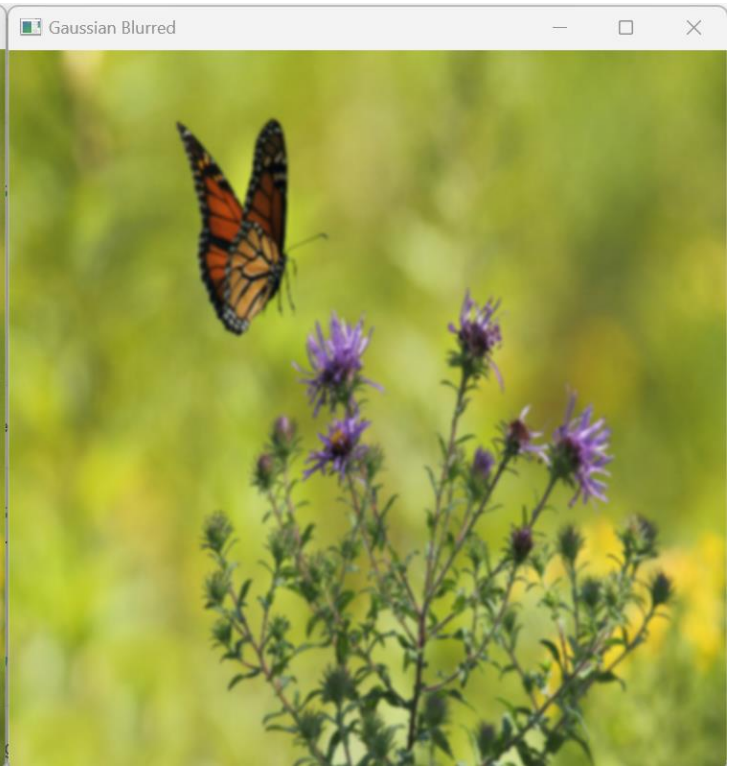
It is generally a trade-off between the degree of blurring and the amount of information that is preserved. The ideal size of the filter will depend on the specifics of the image processing task and the desired outcome.

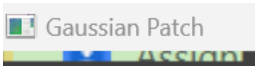
OUTPUT SCREENSHOTS:

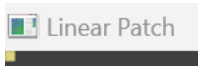


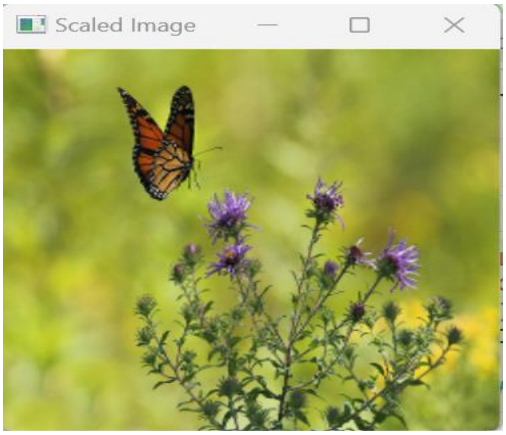


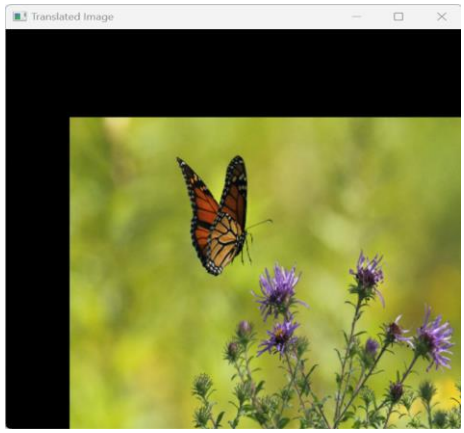

Original

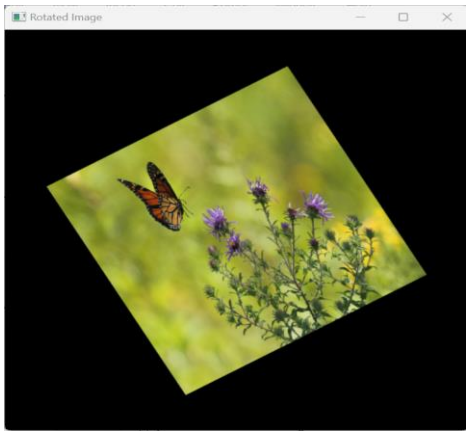

Gaussian Blurred

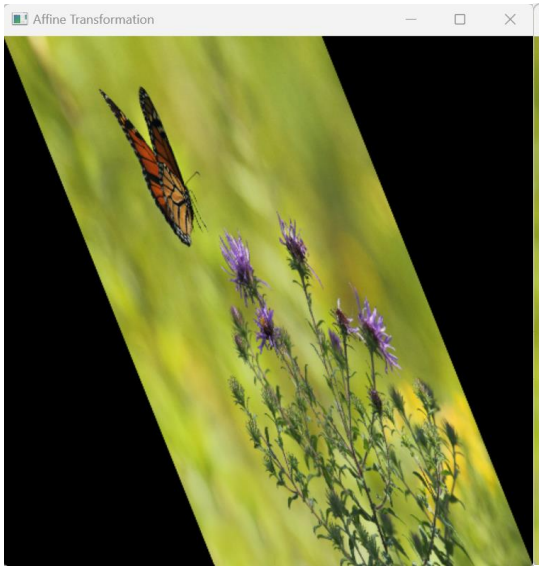

Gaussian Patch

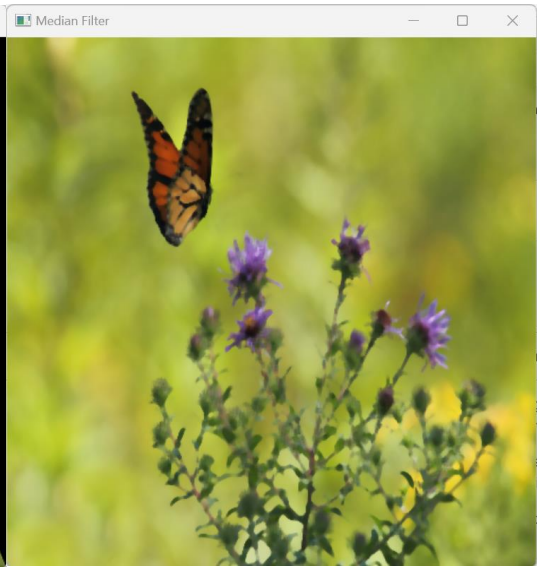

Linear Patch


Scaled Image


Translated Image


Rotated Image


Affine Transformation


Median Filter