# Edge detection using Canny, Prewitt, and Gaussian Blur

### Edge detection is an image processing technique used to identify edges

## How are edges detected?

### Sudden changes in pixel intensity characterize edges.We look for such changes in the neighboring pixels to detect edges

# Canny edge detection

## it is a three stage process for extracting edges from an image

## 1)Noise reduction

## 2)Calculating the Intensity Gradient of the Image

## 3)Suppression of False Edges

## 4)Hysteresis Thresholding

In [9]:
```
pip install opencv-python
```

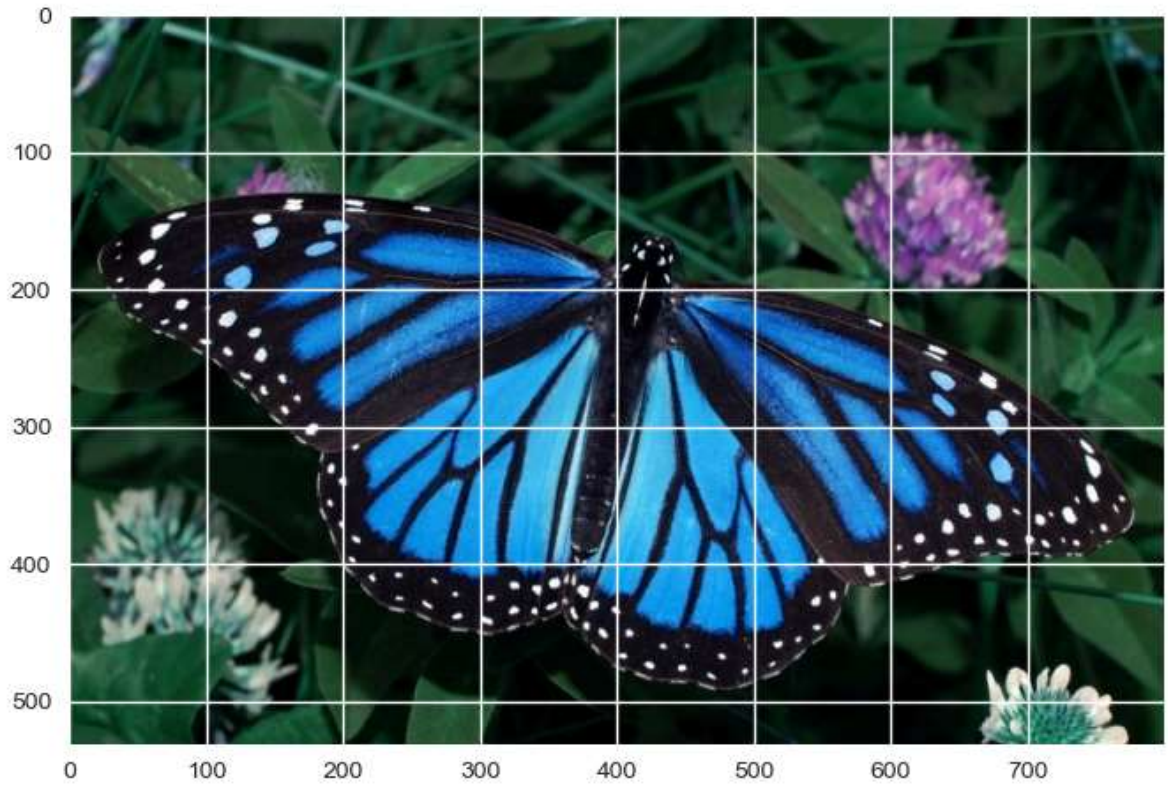
```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: opencv-python in c:\users\kamal\appdata\roamin
g\python\python310\site-packages (4.9.0.80)
Requirement already satisfied: numpy>=1.21.2 in c:\program files\anaconda\lib
\site-packages (from opencv-python) (1.23.5)
Note: you may need to restart the kernel to use updated packages.
```

In [10]:
```
from skimage.io import imread
from matplotlib.pyplot import imshow
from matplotlib.pyplot import plot,subplot
import matplotlib.pyplot as plt
import cv2
import numpy as np
plt.style.use('seaborn')
```

```
C:\Users\kamal\AppData\Local\Temp\ipykernel_9820\1156266310.py:7: MatplotlibD
eprecationWarning: The seaborn styles shipped by Matplotlib are deprecated si
nce 3.6, as they no longer correspond to the styles shipped by seaborn. Howev
er, they will remain available as 'seaborn-v0_8-<style>'. Alternatively, dire
ctly use the seaborn API instead.
  plt.style.use('seaborn')
```

In [11]:
```
butterfly = cv2.imread('butterfly.jpg')
imshow(butterfly)
```

Out[11]:
```
<matplotlib.image.AxesImage at 0x2588993f460>
```




In [12]:
```
image1 = cv2.imread('butterfly.jpg', cv2.IMREAD_GRAYSCALE)
```

## converting image into rgb format

In [13]:
```
image_colour = cv2.cvtColor(image1,cv2.COLOR_BGR2RGB)
```

In [17]:
```
print(image1.dtype)
print(image1.shape)
```

```
uint8
(532, 800)
```

## converting to grayscale

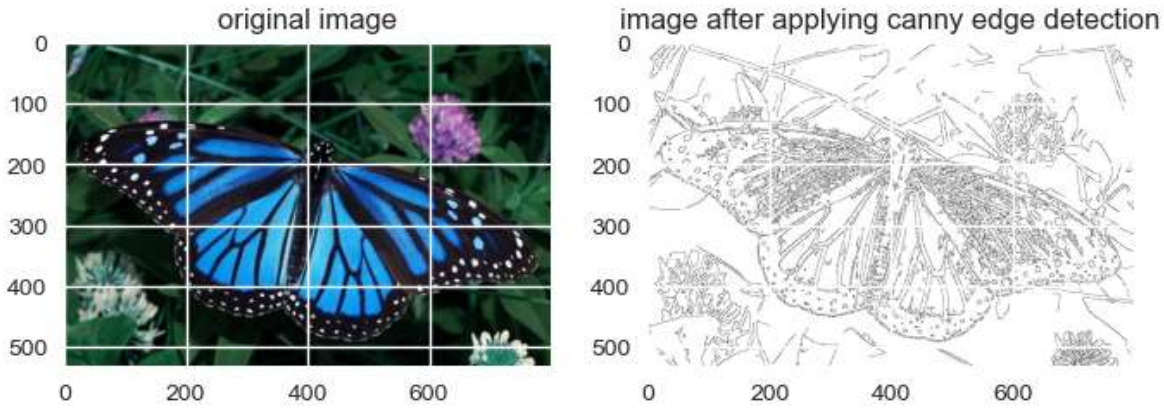
In [22]:
```
gray_image = cv2.cvtColor(butterfly,cv2.COLOR_BGR2GRAY)
```

## applying canny edge detection

In [23]:
```
edged_image = cv2.Canny(gray_image, threshold1 = 30, threshold2 = 100)
```

In [24]:
```
subplot(1,2,1)
imshow(butterfly)
plt.title('original image')

subplot(1,2,2)
imshow(edged_image)
plt.title('image after applying canny edge detection')
```

Out[24]:
```
Text(0.5, 1.0, 'image after applying canny edge detection')
```



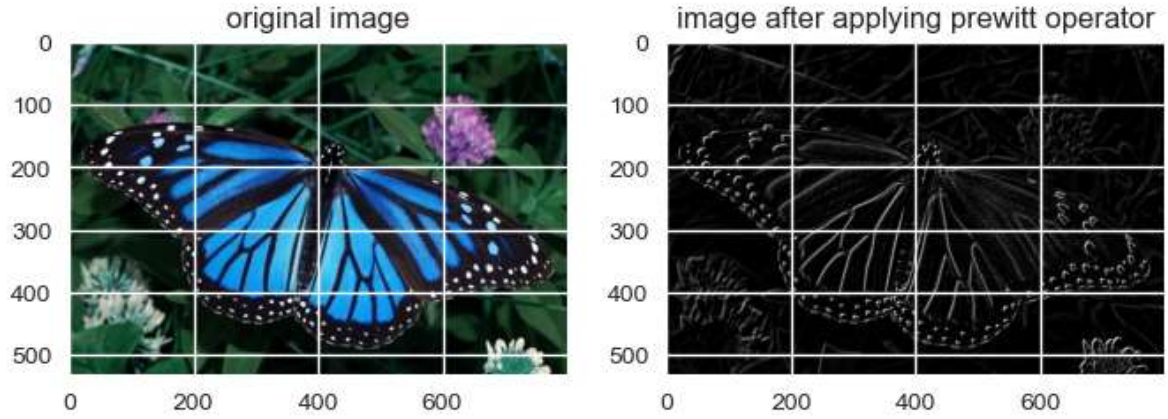

## applying prewitt operator for edge detection

In [25]:
```
prewitt_x = cv2.filter2D(image1, -1, np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0,
prewitt_y = cv2.filter2D(image1, -1, np.array([[-1, -1, -1], [0, 0, 0], [1, 1,
```

## Combining the edge-detected images prewitt_x and prewitt_y

In [26]:
```
prewitt_edges = cv2.addWeighted(prewitt_x, 0.5, prewitt_y, 0.5, 0)
```

In [27]:
```
subplot(1,2,1)
imshow(butterfly)
plt.title('original image')

subplot(1,2,2)
plt.imshow(prewitt_edges, cmap = 'gray')
plt.title('image after applying prewitt operator')
plt.show()
```




## Gaussian blur

## applying gaussian blur to image

In [28]:
```
blurred1 = cv2.GaussianBlur(image1, (5, 5), 0)
blurred2 = cv2.GaussianBlur(image1, (9, 9), 0)
```

## calculating difference of gaussian

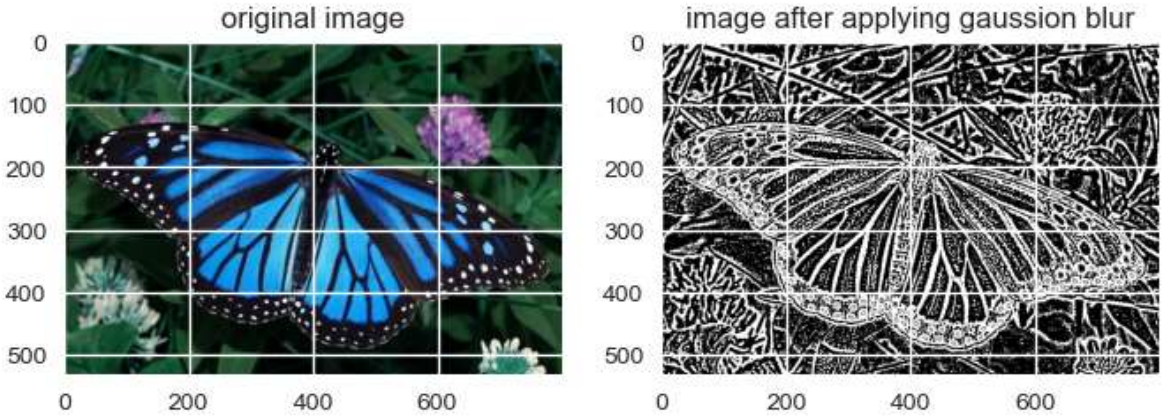
In [29]:
```
dog = blurred1 - blurred2
```

In [ ]:
```

```

# Applying a binary thresholding to the DoG image

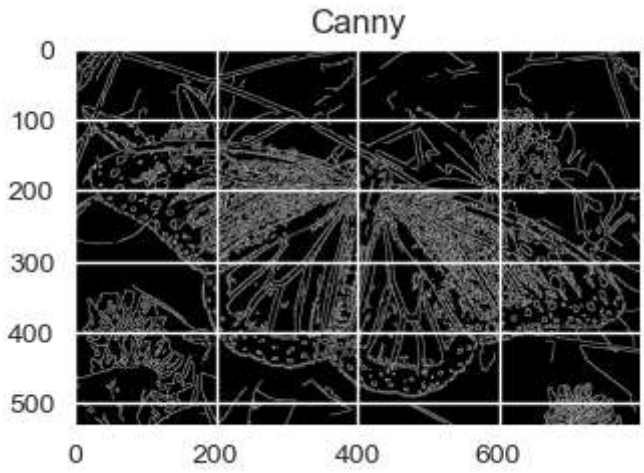
In [30]:
```
_, edges = cv2.threshold(dog, 30, 255, cv2.THRESH_BINARY)
```

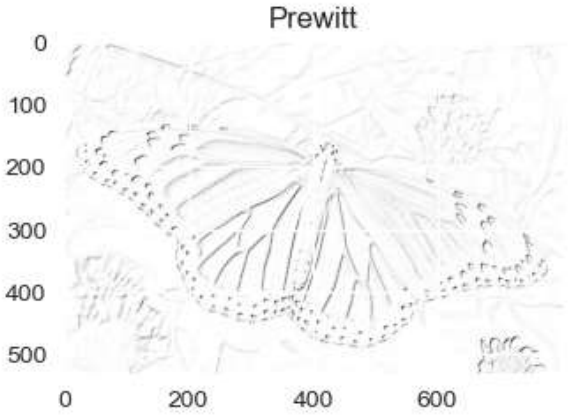
In [31]:
```
subplot(1,2,1)
imshow(butterfly)
plt.title('original image')

subplot(1,2,2)
plt.imshow(edges, cmap='gray')
plt.title('image after applying gaussion blur')
plt.show()
```

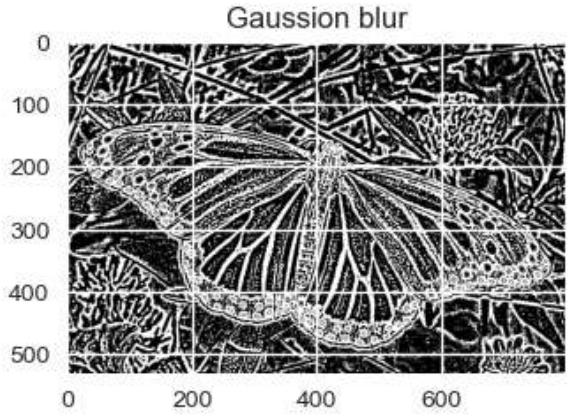



# Comparing all the three image outputs

In [32]:
```
subplot(2,2,1)
plt.imshow(edged_image, cmap='gray')
plt.title('Canny')
plt.show()

subplot(2,2,3)
imshow(prewitt_edges)
plt.title('Prewitt')

subplot(2,2,4)
plt.imshow(edges, cmap='gray')
plt.title('Gaussion blur')
plt.show()
```




In [ ]:
```
```