


```
from google.colab import files
from IPython.display import Image
uploaded=files.upload()
```

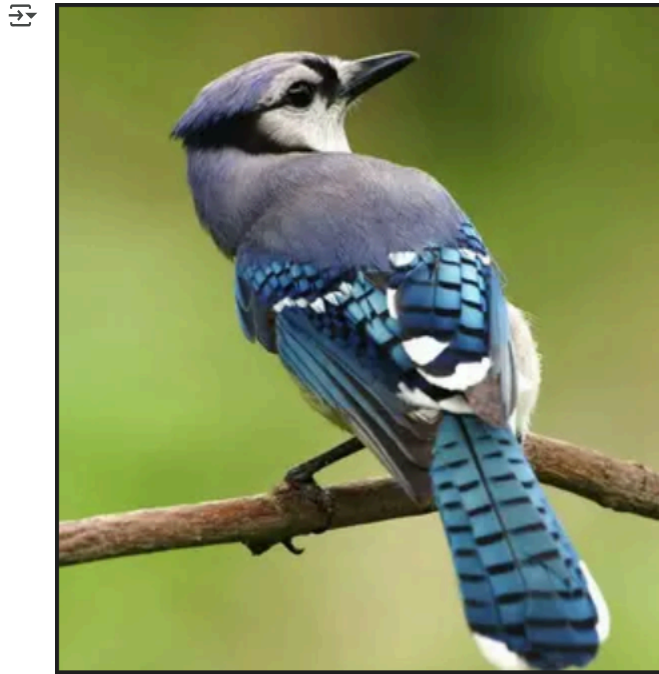
 Choose Files No file chosen  
enable.  
Saving in5.png to in5.png

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to

```
import matplotlib.pyplot as plt
import cv2
from google.colab.patches import cv2_imshow
```

```
image_path = 'ip5.png'
```

```
image=cv2.imread(image_path)
cv2_imshow(image)
```



```
import numpy as np
#Sobel Operator Edge Detection Algorithm
# Check if the image is loaded
if image is None:
    raise ValueError("Error: Image not found. Check the file path.")

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

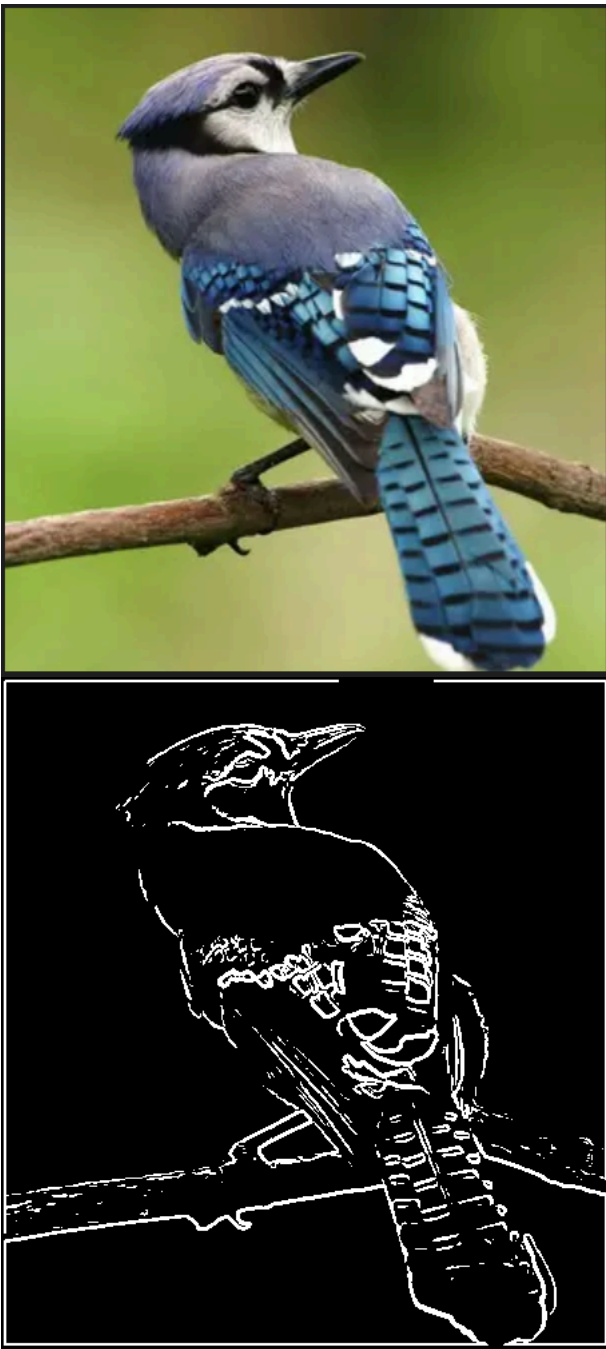
# Apply the Sobel operator to find the x and y gradients
sobel_x = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=3) # Gradient in X direction
sobel_y = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=3) # Gradient in Y direction

# Calculate the magnitude and angle of the gradients
magnitude = np.sqrt(sobel_x**2 + sobel_y**2)
angle = np.arctan2(sobel_y, sobel_x)

# Normalize the magnitude to the range [0, 255]
magnitude = (magnitude / np.max(magnitude) * 255).astype(np.uint8)

# Create a binary image based on the gradient magnitudes
_, binary_image = cv2.threshold(magnitude, 50, 255, cv2.THRESH_BINARY)

# Display the images
cv2_imshow(image) # Original image
cv2_imshow(binary_image) # Edge-detected binary image
```



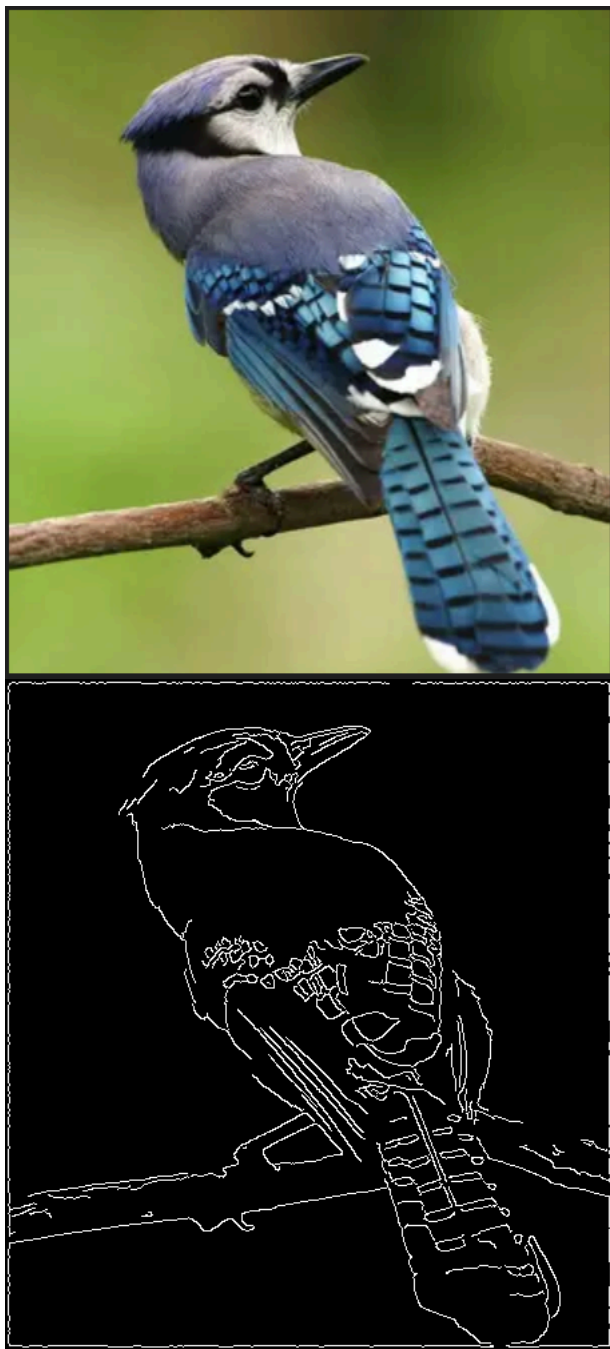
```
#Canny Edge Detection Algorithm
# Check if the image is loaded
if image is None:
    raise ValueError("Error: Image not found. Check the file path.")

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply Gaussian Blur to reduce noise
blurred = cv2.GaussianBlur(gray, (3, 3), 0)

# Use the Canny edge detector to find the edges
edges = cv2.Canny(blurred, 100, 200)

# Display the images
cv2.imshow(image) # Original image
cv2.imshow(edges) # Edge-detected image
```



```
# Laplacian Operator Edge Detection Algorithm
# Check if the image is loaded
if image is None:
    raise ValueError("Error: Image not found. Check the file path.")

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply Gaussian Blur to reduce noise
blurred = cv2.GaussianBlur(gray, (3, 3), 0)

# Use the Laplacian operator to find edges
lap_edges = cv2.Laplacian(blurred, cv2.CV_64F)

# Normalize the edge values to the range [0, 255]
lap_edges_normalized = cv2.normalize(lap_edges, None, 0, 255, cv2.NORM_MINMAX, cv2.CV_8U)

# Display the images
cv2.imshow(image) # Original image
cv2.imshow(lap_edges_normalized) # Laplacian edge-detected image
```



Start coding or [generate](#) with AI.