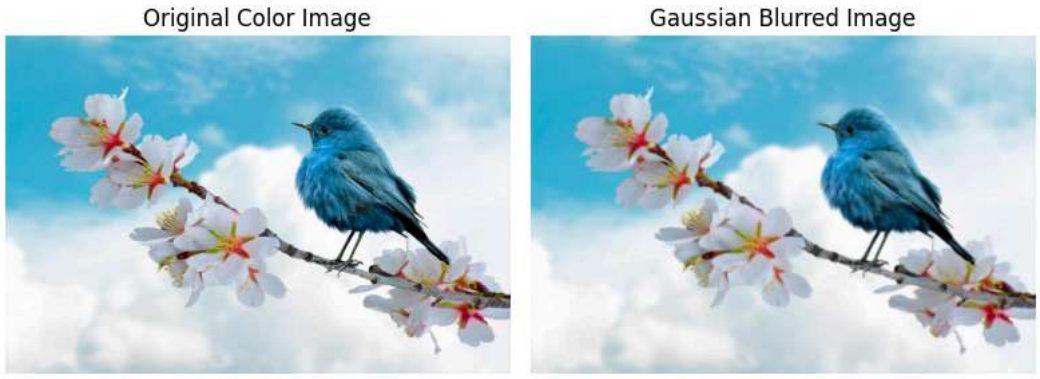
```
from google.colab import files
uploaded = files.upload()
```

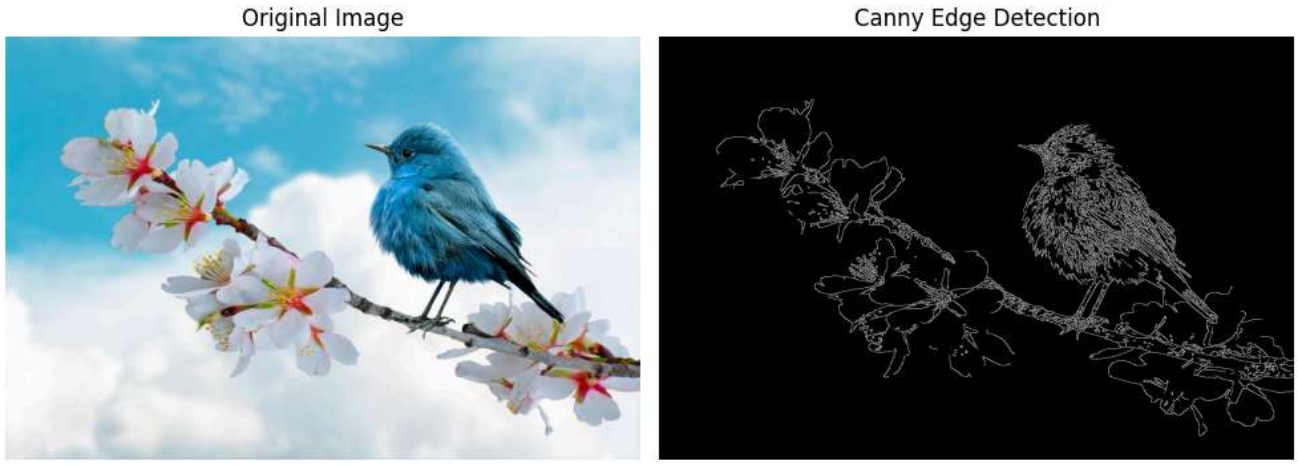
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread("image.ppm")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

plt.imshow(gray, cmap="gray")
plt.axis("off")
plt.show()
```

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread("image.ppm")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

blur = cv2.GaussianBlur(img_rgb, (31 , 31), 1.5)

plt.figure(figsize=(8, 4))

plt.subplot(1, 2, 1)
plt.imshow(img_rgb)
plt.title("Original Color Image")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(blur)
plt.title("Gaussian Blurred Image")
plt.axis("off")

plt.tight_layout()
plt.show()
```




```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread("image.ppm")

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

edges = cv2.Canny(gray, 100, 200)

plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title("Original Image")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(edges, cmap="gray")
plt.title("Canny Edge Detection")
plt.axis("off")

plt.tight_layout()
plt.show()
```




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

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
# Import libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read the image
```

```
img = cv2.imread(list(uploaded.keys())[0])

# Convert to grayscale
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Create a structuring element (kernel)
kernel = np.ones((5,5), np.uint8)

# Apply dilation
dilated_img = cv2.dilate(gray_img, kernel, iterations=1)

# Display original and dilated images
plt.figure(figsize=(10,5))

plt.subplot(1,2,1)
plt.title("Original Image")
plt.imshow(gray_img, cmap='gray')
plt.axis('off')

plt.subplot(1,2,2)
plt.title("Dilated Image")
plt.imshow(dilated_img, cmap='gray')
plt.axis('off')

plt.show()
```


```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread("image.ppm", cv2.IMREAD_GRAYSCALE)


if img is None:
    raise FileNotFoundError("Image not found. Check the filename or path.")

kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))


eroded = cv2.erode(img, kernel, iterations=1)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(img, cmap="gray")
plt.title("Original")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(eroded, cmap="gray")
plt.title("Eroded")
plt.axis("off")

plt.tight_layout()
plt.show()
```


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

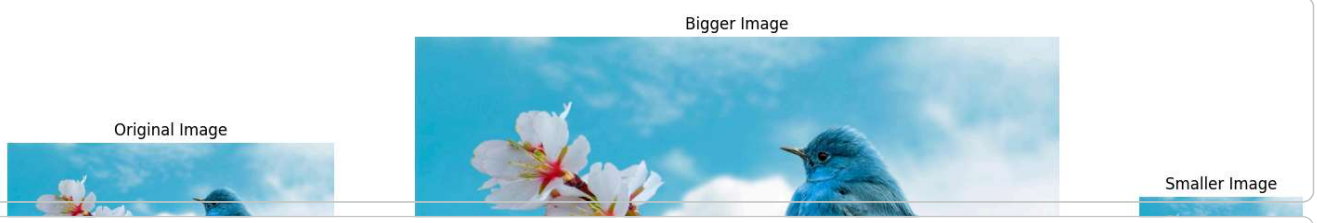
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

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

import cv2
import time
from google.colab.patches import cv2_imshow

video_path = list(uploaded.keys())[0]
cap = cv2.VideoCapture(video_path)


# Fast motion : 0.005
delay = 0.1

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    cv2_imshow(frame)
    time.sleep(delay)

cap.release()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (4).ppm
```

```
from google.colab import files
import cv2
import matplotlib.pyplot as plt

import cv2
import matplotlib.pyplot as plt


img = cv2.imread(list(uploaded.keys())[0])


bigger_img = cv2.resize(img, None, fx=2, fy=2, interpolation=cv2.INTER_LINEAR)
smaller_img = cv2.resize(img, None, fx=0.5, fy=0.5, interpolation=cv2.INTER_AREA)


fig, axs = plt.subplots(
    1, 3,
    figsize=(18, 6),
    gridspec_kw={'width_ratios': [1, 2, 0.5]}
)

axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axs[0].set_title("Original Image")
axs[0].axis('off')

axs[1].imshow(cv2.cvtColor(bigger_img, cv2.COLOR_BGR2RGB))
axs[1].set_title("Bigger Image")
axs[1].axis('off')

axs[2].imshow(cv2.cvtColor(smaller_img, cv2.COLOR_BGR2RGB))
axs[2].set_title("Smaller Image")
axs[2].axis('off')

plt.show()
```




```
from google.colab import files
uploaded = files.upload()
from PIL import Image
import matplotlib.pyplot as plt


image_name = list(uploaded.keys())[0]


img = Image.open(image_name)


clockwise = img.rotate(-90, expand=True)
counter_clockwise = img.rotate(90, expand=True)


plt.figure(figsize=(12,4))

plt.subplot(1,3,1)
plt.title("Original")
plt.imshow(img)
plt.axis("off")

plt.subplot(1,3,2)
plt.title("Clockwise")
plt.imshow(clockwise)
plt.axis("off")

plt.subplot(1,3,3)
plt.title("Counter Clockwise")
plt.imshow(counter_clockwise)
plt.axis("off")

plt.show()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (5).ppm
```


```
from google.colab import files
uploaded = files.upload()
from PIL import Image
import matplotlib.pyplot as plt


img_name = list(uploaded.keys())[0]
img = Image.open(img_name)
```

```
width, height = img.size


output = Image.new("RGB", (width, height), (255, 255, 255))


tx = 100
ty = 60


output.paste(img, (tx, ty))


plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.title("Input Image")
plt.imshow(img)
plt.axis("off")

plt.subplot(1,2,2)
plt.title("Moved Image")
plt.imshow(output)
plt.axis("off")

plt.show()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (6).ppm
```


```
from google.colab import files
uploaded = files.upload()
import cv2
import numpy as np
import matplotlib.pyplot as plt


img_name = list(uploaded.keys())[0]


img = cv2.imread(img_name)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)


rows, cols, ch = img.shape

src_points = np.float32([
    [0, 0],
    [cols - 1, 0],
    [0, rows - 1]
])


dst_points = np.float32([
    [50, 50],
    [cols - 100, 80],
    [80, rows - 100]
])


matrix = cv2.getAffineTransform(src_points, dst_points)
```

```
affine_img = cv2.warpAffine(img, matrix, (cols, rows))


plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.title("Original Image")
plt.imshow(img)
plt.axis("off")

plt.subplot(1,2,2)
plt.title("Affine Transformed Image")
plt.imshow(affine_img)
plt.axis("off")

plt.show()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (7).ppm
```


```
from google.colab import files
uploaded = files.upload()
import cv2
import numpy as np
import matplotlib.pyplot as plt


img_name = list(uploaded.keys())[0]


img = cv2.imread(img_name)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)


rows, cols, ch = img.shape

src_points = np.float32([
    [0, 0],
    [cols - 1, 0],
    [cols - 1, rows - 1],
    [0, rows - 1]
])

dst_points = np.float32([
    [50, 50],
    [cols - 100, 30],
    [cols - 50, rows - 50],
    [100, rows - 80]
])


matrix = cv2.getPerspectiveTransform(src_points, dst_points)


perspective_img = cv2.warpPerspective(img, matrix, (cols, rows))


plt.figure(figsize=(10,4))
```

```
plt.subplot(1,2,1)
plt.title("Original Image")
plt.imshow(img)
plt.axis("off")

plt.subplot(1,2,2)
plt.title("Perspective Transformed Image")
plt.imshow(perspective_img)
plt.axis("off")

plt.show()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (8).ppm
```


```
from google.colab import files
uploaded = files.upload()
import cv2
import numpy as np
import matplotlib.pyplot as plt

img_name = list(uploaded.keys())[0]


img = cv2.imread(img_name)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)


rows, cols, ch = img.shape


src_points = np.float32([
    [0, 0],
    [cols - 1, 0],
    [cols - 1, rows - 1],
    [0, rows - 1]
])


dst_points = np.float32([
    [80, 60],
    [cols - 120, 40],
    [cols - 60, rows - 80],
    [100, rows - 100]
])


H, status = cv2.findHomography(src_points, dst_points)


homography_img = cv2.warpPerspective(img, H, (cols, rows))


plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.title("Original Image")
plt.imshow(img)
plt.axis("off")

plt.subplot(1,2,2)
plt.title("Homography Transformed Image")
```

```
plt.title("Homography Transformed Image")
plt.imshow(homography_img)
plt.axis("off")

plt.show()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (9).ppm
```


```
from google.colab import files
uploaded = files.upload()
import cv2
import numpy as np
import matplotlib.pyplot as plt


img_name = list(uploaded.keys())[0]


img = cv2.imread(img_name)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
rows, cols, ch = img.shape


src_pts = np.array([
    [0, 0],
    [cols-1, 0],
    [cols-1, rows-1],
    [0, rows-1]
], dtype=np.float32)


dst_pts = np.array([
    [80, 60],
    [cols-120, 40],
    [cols-60, rows-80],
    [100, rows-100]
], dtype=np.float32)


def compute_homography(src, dst):
    A = []
    for i in range(len(src)):
        x, y = src[i][0], src[i][1]
        u, v = dst[i][0], dst[i][1]
        A.append([-x, -y, -1, 0, 0, 0, x*u, y*u, u])
        A.append([0, 0, 0, -x, -y, -1, x*v, y*v, v])
    A = np.array(A)

    U, S, Vt = np.linalg.svd(A)
    H = Vt[-1].reshape(3,3)
    return H / H[2,2]

H = compute_homography(src_pts, dst_pts)

dlt_img = cv2.warpPerspective(img, H, (cols, rows))


plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.title("Original Image")
plt.imshow(img)
```

```
plt.axis("off")

plt.subplot(1,2,2)
plt.title("DLT Transformed Image")
plt.imshow(dlt_img)
plt.axis("off")
plt.show()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (10).ppm
```

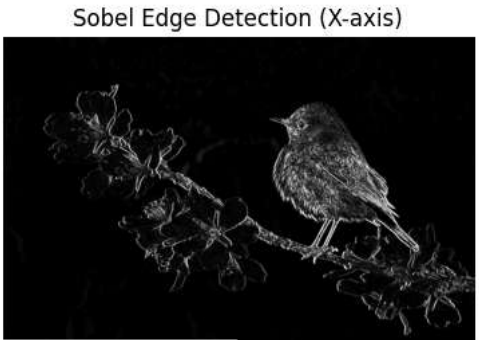

```
from google.colab import files
uploaded = files.upload()
import cv2
import matplotlib.pyplot as plt


img_name = list(uploaded.keys())[0]

img = cv2.imread(img_name, cv2.IMREAD_GRAYSCALE)


blur = cv2.GaussianBlur(img, (5,5), 1.4)


edges = cv2.Canny(blur, threshold1=50, threshold2=150)


plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.title("Original Grayscale Image")
plt.imshow(img, cmap='gray')
plt.axis("off")

plt.subplot(1,2,2)
plt.title("Canny Edge Detection")
plt.imshow(edges, cmap='gray')
plt.axis("off")

plt.show()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (11).ppm
```

```
from google.colab import files
uploaded = files.upload()
import cv2
import matplotlib.pyplot as plt
import numpy as np


img_name = list(uploaded.keys())[0]

img = cv2.imread(img_name, cv2.IMREAD_GRAYSCALE)

sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)

sobelx = cv2.convertScaleAbs(sobelx)


plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.title("Original Grayscale Image")
plt.imshow(img, cmap='gray')
plt.axis("off")

plt.subplot(1,2,2)
plt.title("Sobel Edge Detection (X-axis)")
plt.imshow(sobelx, cmap='gray')
plt.axis("off")

plt.show()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving image.ppm to image (12).ppm
```

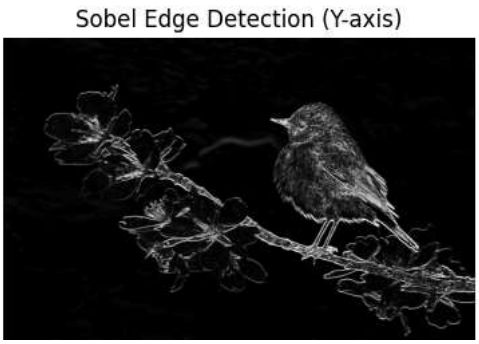



```
from google.colab import files
uploaded = files.upload()
import cv2
import matplotlib.pyplot as plt
import numpy as np

img_name = list(uploaded.keys())[0]


img = cv2.imread(img_name, cv2.IMREAD_GRAYSCALE)


sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)

sobely = cv2.convertScaleAbs(sobely)

# Display images
plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.title("Original Grayscale Image")
plt.imshow(img, cmap='gray')
plt.axis("off")

plt.subplot(1,2,2)
plt.title("Sobel Edge Detection (Y-axis)")
plt.imshow(sobely, cmap='gray')
plt.axis("off")
```

```
plt.show()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (13).ppm
```



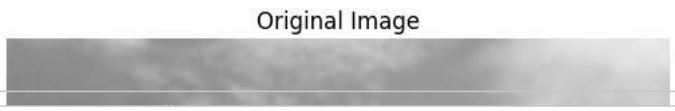

```
from google.colab import files
uploaded = files.upload()
import cv2
import matplotlib.pyplot as plt
import numpy as np


img_name = list(uploaded.keys())[0]


img = cv2.imread(img_name, cv2.IMREAD_GRAYSCALE)


sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
sobelx = cv2.convertScaleAbs(sobelx)

sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
sobely = cv2.convertScaleAbs(sobely)


sobelxy = cv2.addWeighted(sobelx, 0.5, sobely, 0.5, 0)


plt.figure(figsize=(15,4))

plt.subplot(1,4,1)
plt.title("Original Image")
plt.imshow(img, cmap='gray')
plt.axis("off")

plt.subplot(1,4,2)
plt.title("Sobel X")
plt.imshow(sobelx, cmap='gray')
plt.axis("off")

plt.subplot(1,4,3)
plt.title("Sobel Y")
plt.imshow(sobely, cmap='gray')
plt.axis("off")

plt.subplot(1,4,4)
plt.title("Sobel XY")
plt.imshow(sobelxy, cmap='gray')
plt.axis("off")

plt.show()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving image.ppm to image (14).ppm
```


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

import cv2
import numpy as np
from matplotlib import pyplot as plt


image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

plt.imshow(img, cmap='gray')
plt.title("Original Image")
plt.axis('off')

laplacian_mask = np.array([[0, 1, 0],
                           [1, -4, 1],
                           [0, 1, 0]])


laplacian_img = cv2.filter2D(img, -1, laplacian_mask)

sharpened_img = cv2.subtract(img, laplacian_img)


plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.imshow(laplacian_img, cmap='gray')
plt.title("Laplacian Image")
plt.axis('off')

plt.subplot(1,2,2)
plt.imshow(sharpened_img, cmap='gray')
plt.title("Sharpened Image")
plt.axis('off')
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (15).ppm
(np.float64(-0.5), np.float64(1279.5), np.float64(852.5), np.float64(-0.5))
```




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


import cv2
import numpy as np
from matplotlib import pyplot as plt


image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)


plt.imshow(img, cmap='gray')
plt.title("Original Image")
plt.axis('off')

laplacian_mask = np.array([[1, 1, 1],
                           [1, -8, 1],
                           [1, 1, 1]])

laplacian_img = cv2.filter2D(img, -1, laplacian_mask)

sharpened_img = cv2.subtract(img, laplacian_img)


plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.imshow(laplacian_img, cmap='gray')
plt.title("Laplacian Image")
plt.axis('off')

plt.subplot(1,2,2)
plt.imshow(sharpened_img, cmap='gray')
plt.title("Sharpened Image")
plt.axis('off')
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving image.ppm to image (16).ppm
(np.float64(-0.5), np.float64(1279.5), np.float64(852.5), np.float64(-0.5))
```


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


import cv2
import numpy as np
from matplotlib import pyplot as plt


image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

plt.imshow(img, cmap='gray')
plt.title("Original Image")
plt.axis('off')

laplacian_mask = np.array([[0, -1, 0],
                           [-1, 5, -1],
                           [0, -1, 0]])

sharpened_img = cv2.filter2D(img, -1, laplacian_mask)


plt.figure(figsize=(5,4))
plt.imshow(sharpened_img, cmap='gray')
plt.title("Sharpened Image (Positive Center Laplacian)")
plt.axis('off')
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (17).ppm
(np.float64(-0.5), np.float64(1279.5), np.float64(852.5), np.float64(-0.5))
```

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


import cv2
import numpy as np
from matplotlib import pyplot as plt

image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

plt.imshow(img, cmap='gray')
plt.title("Original Image")
plt.axis('off')

blurred_img = cv2.GaussianBlur(img, (5,5), 0)


sharpened_img = cv2.subtract(img, blurred_img)

plt.figure(figsize=(12,4))

plt.subplot(1,3,1)
plt.imshow(img, cmap='gray')
plt.title("Original Image")
plt.axis('off')

plt.subplot(1,3,2)
plt.imshow(blurred_img, cmap='gray')
plt.title("Blurred Image")
plt.axis('off')

plt.subplot(1,3,3)
plt.imshow(sharpened_img, cmap='gray')
plt.title("Sharpened Image (Unsharp Masking)")
plt.axis('off')
```

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


import cv2
import numpy as np
from matplotlib import pyplot as plt


image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

plt.imshow(img, cmap='gray')
plt.title("Original Image")
plt.axis('off')

A = 2

high_boost_mask = np.array([[0, -1, 0],
                            [-1, A+4, -1],
                            [0, -1, 0]])

sharpened_img = cv2.filter2D(img, -1, high_boost_mask)

plt.figure(figsize=(5,4))
plt.imshow(sharpened_img, cmap='gray')
plt.title("High-Boost Sharpened Image")
plt.axis('off')
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving image.ppm to image (19).ppm
(np.float64(-0.5), np.float64(1279.5), np.float64(852.5), np.float64(-0.5))
```


```
from google.colab import files
```

```
uploaded = files.upload()


import cv2
import numpy as np
from matplotlib import pyplot as plt

image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)


plt.imshow(img, cmap='gray')
plt.title("Original Image")
plt.axis('off')

Gx = np.array([[-1, 0, 1],
               [-2, 0, 2],
               [-1, 0, 1]])

Gy = np.array([[-1, -2, -1],
               [ 0,  0,  0],
               [ 1,  2,  1]])


grad_x = cv2.filter2D(img, -1, Gx)
grad_y = cv2.filter2D(img, -1, Gy)

gradient = cv2.add(np.abs(grad_x), np.abs(grad_y))


sharpened_img = cv2.add(img, gradient)


plt.figure(figsize=(12,4))

plt.subplot(1,3,1)
plt.imshow(gradient, cmap='gray')
plt.title("Gradient Image")
plt.axis('off')

plt.subplot(1,3,2)
plt.imshow(sharpened_img, cmap='gray')
plt.title("Sharpened Image")
plt.axis('off')
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving image.ppm to image (20).ppm
(np.float64(-0.5), np.float64(1279.5), np.float64(852.5), np.float64(-0.5))

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

import cv2
import numpy as np
from matplotlib import pyplot as plt


image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path)

watermarked_img = img.copy()
text = "WATERMARK"
position = (50, 50)

cv2.putText(watermarked_img, text, position,
            cv2.FONT_HERSHEY_SIMPLEX,
            1, (255, 255, 255), 2, cv2.LINE_AA)

plt.imshow(cv2.cvtColor(watermarked_img, cv2.COLOR_BGR2RGB))
plt.title("Watermarked Image")
plt.axis('off')
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving image.ppm to image (21).ppm
(np.float64(-0.5), np.float64(1279.5), np.float64(852.5), np.float64(-0.5))


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

import cv2
import numpy as np
from matplotlib import pyplot as plt

image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)


_, binary = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)


kernel = np.ones((3,3), np.uint8)


eroded = cv2.erode(binary, kernel, iterations=1)

boundary = cv2.subtract(binary, eroded)
```

```
plt.figure(figsize=(12,4))

plt.subplot(1,3,1)
plt.imshow(binary, cmap='gray')
plt.title("Binary Image")
plt.axis('off')

plt.subplot(1,3,2)
plt.imshow(eroded, cmap='gray')
plt.title("Eroded Image")
plt.axis('off')

plt.subplot(1,3,3)
plt.imshow(boundary, cmap='gray')
plt.title("Boundary Image")
plt.axis('off')
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

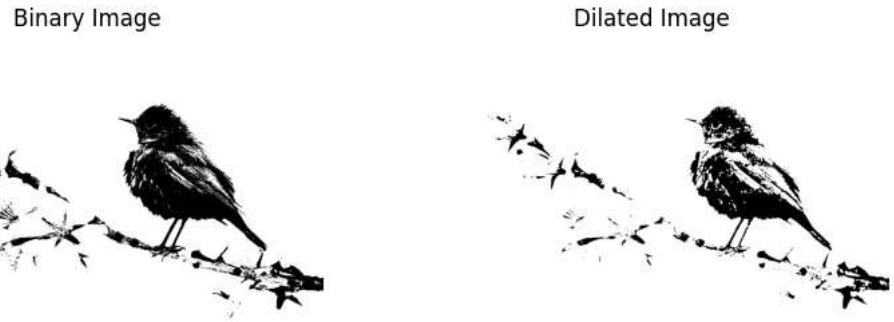
```
Saving image.ppm to image (22).ppm
(np.float64(-0.5), np.float64(1279.5), np.float64(852.5), np.float64(-0.5))
```


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

import cv2
import numpy as np
from matplotlib import pyplot as plt

image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

_, binary = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

kernel = np.ones((3,3), np.uint8)
eroded = cv2.erode(binary, kernel, iterations=1)

plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.imshow(binary, cmap='gray')
plt.title("Binary Image")
plt.axis('off')

plt.subplot(1,2,2)
plt.imshow(eroded, cmap='gray')
plt.title("Eroded Image")
plt.axis('off')
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving image.ppm to image (23).ppm
(np.float64(-0.5), np.float64(1279.5), np.float64(852.5), np.float64(-0.5))


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

import cv2
import numpy as np
from matplotlib import pyplot as plt

image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

_, binary = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

kernel = np.ones((3,3), np.uint8)
dilated = cv2.dilate(binary, kernel, iterations=1)

plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.imshow(binary, cmap='gray')
plt.title("Binary Image")
plt.axis('off')

plt.subplot(1,2,2)
plt.imshow(dilated, cmap='gray')
plt.title("Dilated Image")
plt.axis('off')
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving image.ppm to image (24).ppm
(np.float64(-0.5), np.float64(1279.5), np.float64(852.5), np.float64(-0.5))




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

import cv2
import numpy as np
from matplotlib import pyplot as plt

image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

_, binary = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

kernel = np.ones((3,3), np.uint8)
opening = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel)

plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.imshow(binary, cmap='gray')
plt.title("Binary Image")
```

```
plt.axis('off')

plt.subplot(1,2,2)
plt.imshow(opening, cmap='gray')
plt.title("Opening Result")
plt.axis('off')
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (25).ppm
(np.float64(-0.5), np.float64(1279.5), np.float64(852.5), np.float64(-0.5))
```


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

import cv2
import numpy as np
from matplotlib import pyplot as plt

image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

_, binary = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

kernel = np.ones((3,3), np.uint8)
closing = cv2.morphologyEx(binary, cv2.MORPH_CLOSE, kernel)

plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.imshow(binary, cmap='gray')
plt.title("Binary Image")
plt.axis('off')

plt.subplot(1,2,2)
plt.imshow(closing, cmap='gray')
plt.title("Closing Result")
plt.axis('off')
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (26).ppm
(np.float64(-0.5), np.float64(1279.5), np.float64(852.5), np.float64(-0.5))
```

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

import cv2
import numpy as np
from matplotlib import pyplot as plt

image_path = list(uploaded.keys())[0]
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

_, binary = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

kernel = np.ones((3,3), np.uint8)
gradient = cv2.morphologyEx(binary, cv2.MORPH_GRADIENT, kernel)

plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.imshow(binary, cmap='gray')
plt.title("Binary Image")
plt.axis('off')

plt.subplot(1,2,2)
plt.imshow(gradient, cmap='gray')
plt.title("Morphological Gradient")
plt.axis('off')
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image.ppm to image (27).ppm
(np.float64(-0.5), np.float64(1279.5), np.float64(852.5), np.float64(-0.5))
```