Filters and Edge Detection

1.Sobel Filter

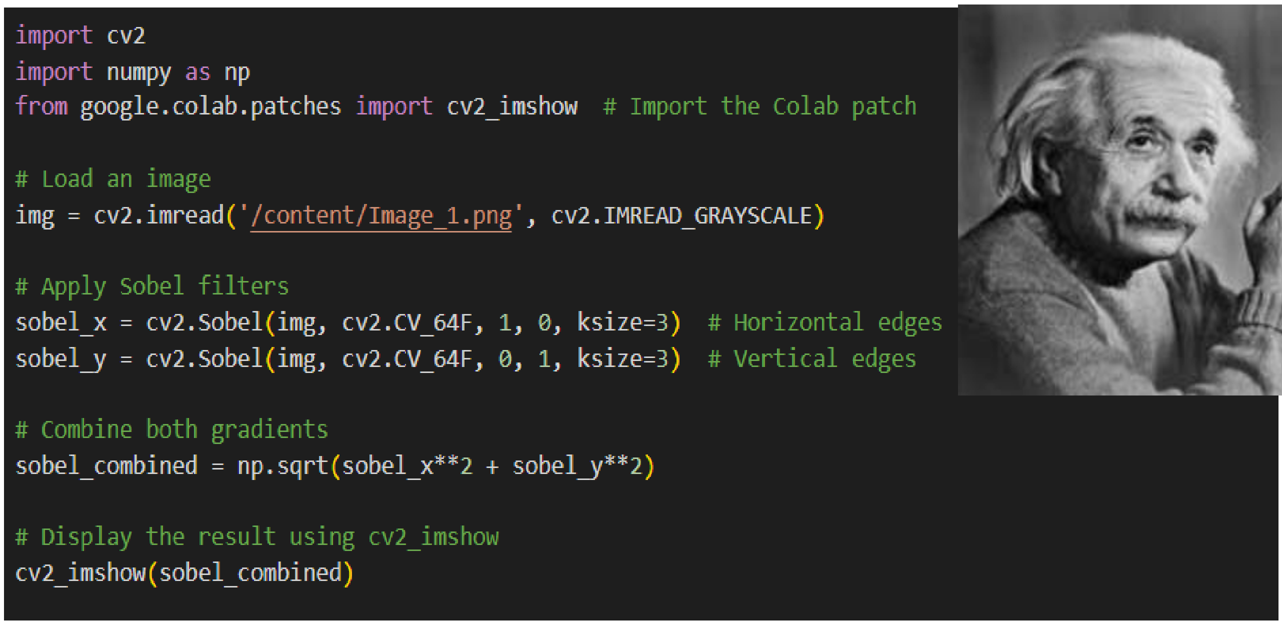
1.1 Working Principle

By measuring an image's 2-D spatial gradient, the Sobel operator highlights areas of high spatial frequency that line up with edges. It is typically applied to an input grayscale image to get the approximate absolute gradient magnitude at each location. It is determined by a specific matrix or window size, such as 3x3 or 5x5.

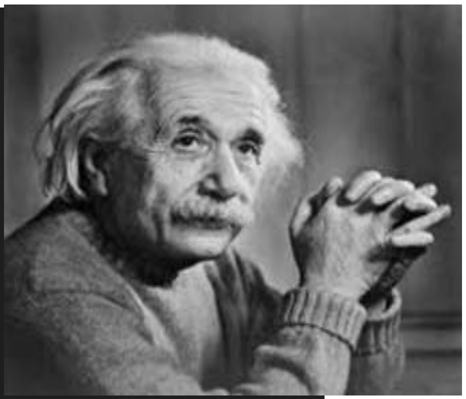
1.2 Usage

Applications including image segmentation, feature extraction, and computer vision edge detection often use the Sobel filter.

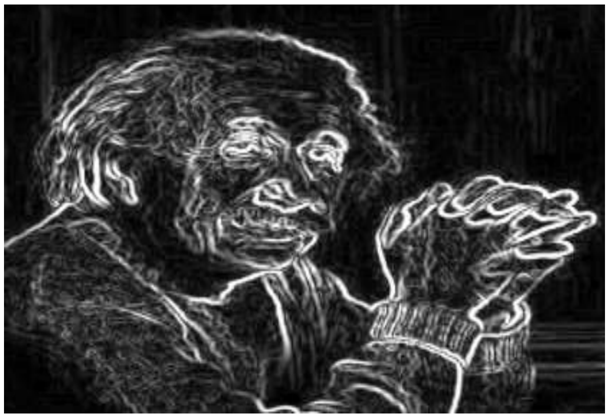
By using the following image and applying the Sobel filter code



```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow  # Import the Colab patch

# Load an image
img = cv2.imread('/content/Image_1.png', cv2.IMREAD_GRAYSCALE)

# Apply Sobel filters
sobel_x = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)  # Horizontal edges
sobel_y = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)  # Vertical edges

# Combine both gradients
sobel_combined = np.sqrt(sobel_x**2 + sobel_y**2)

# Display the result using cv2_imshow
cv2_imshow(sobel_combined)
```



The output:



Sobel filter output

2. Laplacian Filter

2.1 Working Principle

A representation of a second order or second derivative technique of enhancement is the Laplacian of an image, which identifies areas of rapid intensity shift. It is particularly effective at identifying an image's minute elements. An operator with Laplacian shape will improve any feature that has a sharp discontinuity.

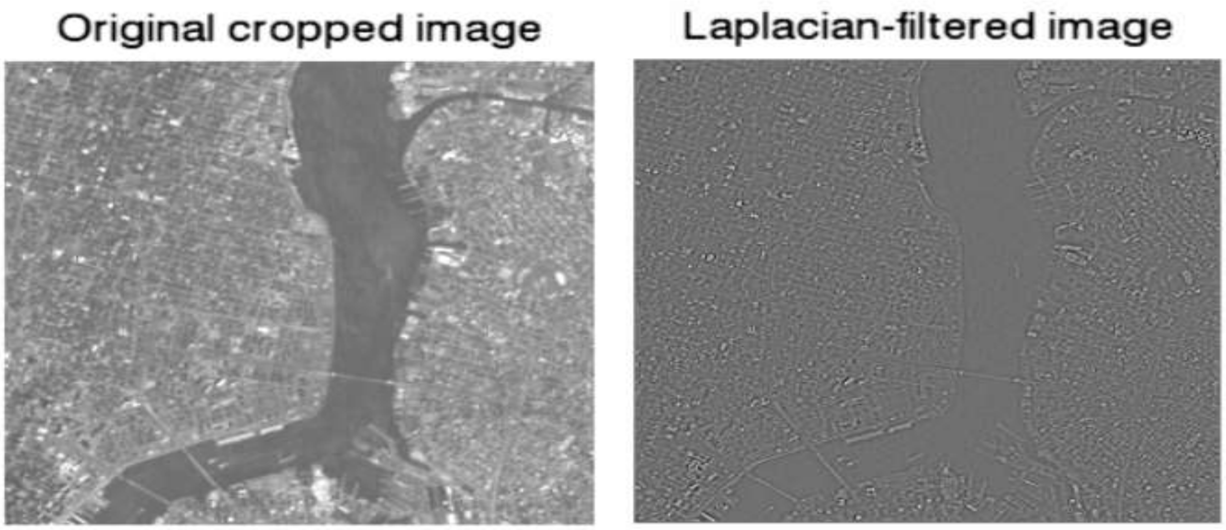
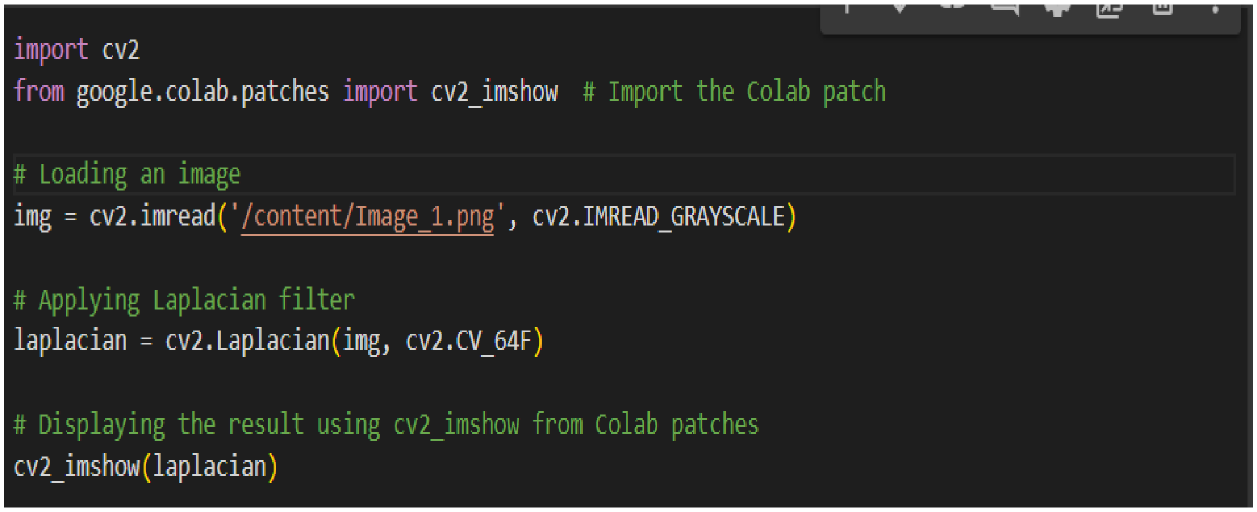


Figure 2.1

2.2 Usage

Since the Laplacian filter is isotropic, which edges of any orientation can be detected. This makes it especially useful for preprocessing stages of more complicated image analysis jobs, as well as for applications like noise reduction and image sharpening.



```
import cv2
from google.colab.patches import cv2_imshow  # Import the Colab patch

# Loading an image
img = cv2.imread('/content/Image_1.png', cv2.IMREAD_GRAYSCALE)

# Applying Laplacian filter
laplacian = cv2.Laplacian(img, cv2.CV_64F)

# Displaying the result using cv2_imshow from Colab patches
cv2_imshow(laplacian)
```

The output:

Laplacian filter output

3. Canny Edge Detector

3.1 Working Principle

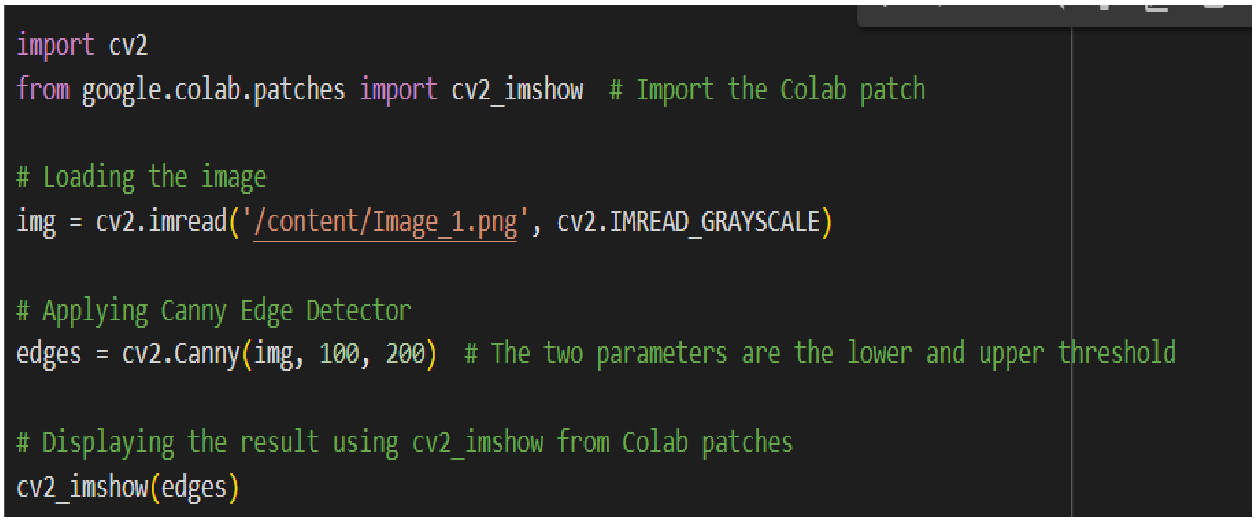
There are several steps involved in the Canny operator's job. The image is first smoothed using Gaussian convolution. Next, the smoothed image is subjected to a basic 2-D first derivative operator in order to identify parts of the image that have high first spatial derivatives.

In the image with gradient magnitude, edges give rise to ridges. The technique known as non-maximal suppression is where the algorithm tracks along the top of these ridges and sets to zero those pixels that are not actually on the ridge top to produce a thin line in the output.

Hysteresis in the tracking process is regulated by two thresholds, T1 and T2, where T1 is greater than T2. Only at a location on a ridge above T1 can tracking start. After that, tracking is carried out in both directions until the ridge's height drops below T2. Hysteresis like this makes sure that noisy edges don't split into several edge fragments.

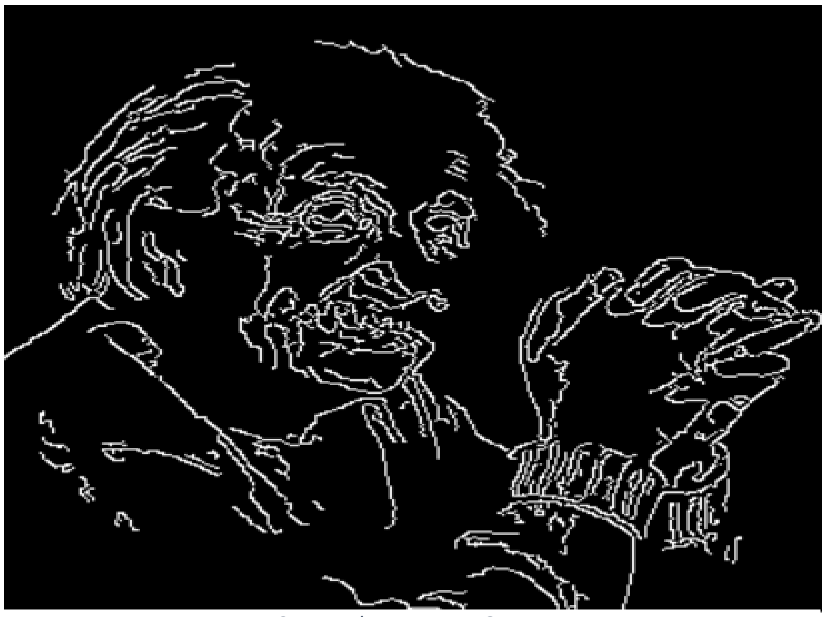
3.2 Usage

In computer vision, this method is frequently used for object detection, image segmentation, and as a preliminary step before additional image analysis.



```
import cv2
from google.colab.patches import cv2_imshow  # Import the Colab patch

# Loading the image
img = cv2.imread('/content/Image_1.png', cv2.IMREAD_GRAYSCALE)

# Applying Canny Edge Detector
edges = cv2.Canny(img, 100, 200)  # The two parameters are the lower and upper threshold

# Displaying the result using cv2_imshow from Colab patches
cv2_imshow(edges)
```

The output:


Canny Edge Detector Output

4. Contours in Image processing

4.1 Working Principle

Contours aid in improved compression by highlighting pixel differences. When choosing which pixels to remove, this is helpful.

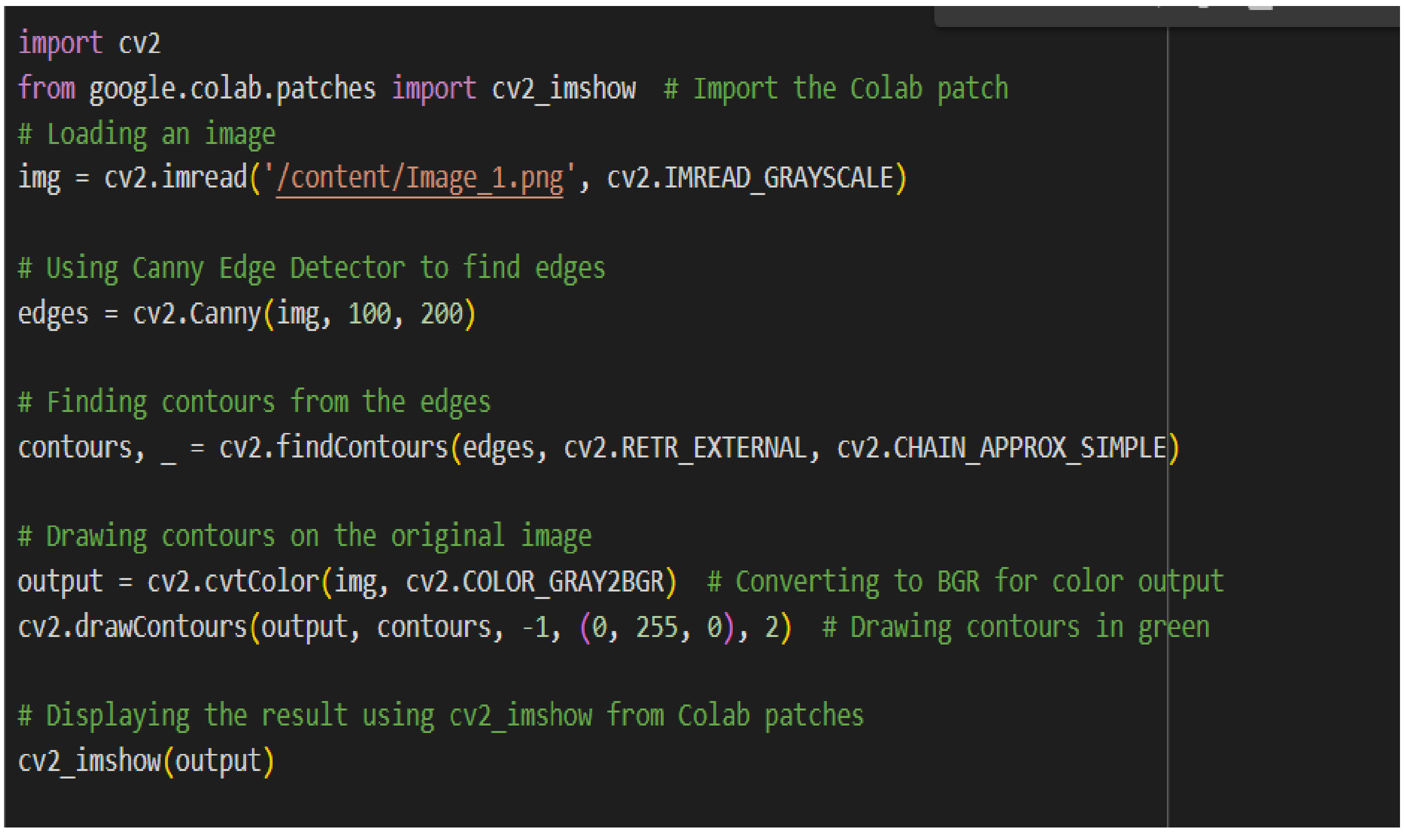
Additionally, contours help algorithms that need specific features in the input image in order to process it. That is the case, for example, if we want to utilize an object's presence as a feature. Checking if the detected contours match those of target items is necessary to determine whether an image contains something.

4.2 Usage

Contours assist boundary point-finding methods used in corner identification.

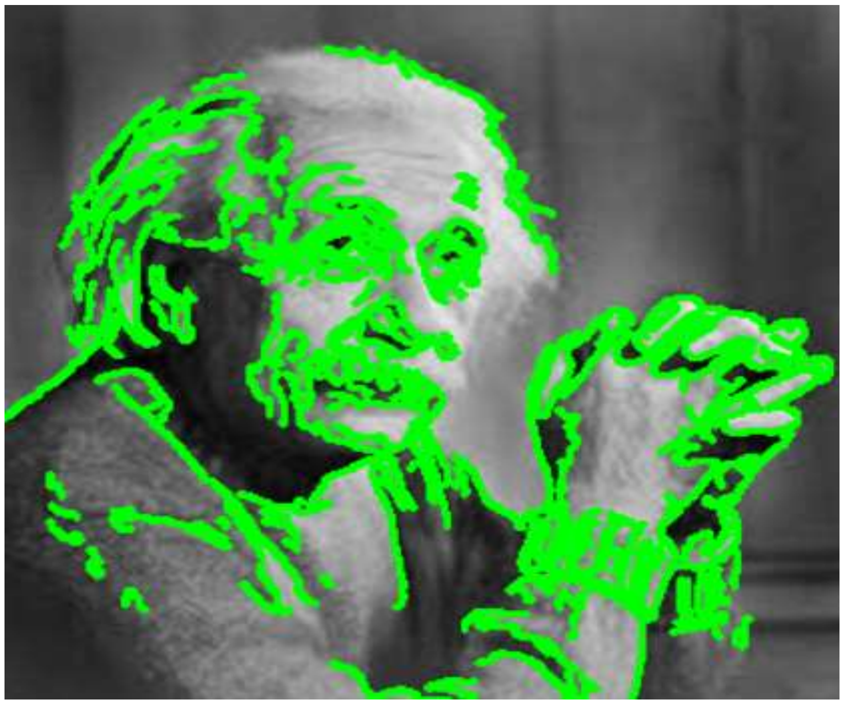
Applications for detecting and segmenting objects in a picture depend heavily on contour detection.

The pixels that make up the border of an object form a contour and these pixels can be distinguished from the others by their common color.



```python
import cv2
from google.colab.patches import cv2_imshow  # Import the Colab patch
# Loading an image
img = cv2.imread('/content/Image_1.png', cv2.IMREAD_GRAYSCALE)

# Using Canny Edge Detector to find edges
edges = cv2.Canny(img, 100, 200)

# Finding contours from the edges
contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Drawing contours on the original image
output = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)  # Converting to BGR for color output
cv2.drawContours(output, contours, -1, (0, 255, 0), 2)  # Drawing contours in green

# Displaying the result using cv2_imshow from Colab patches
cv2_imshow(output)
```

The output:



Contours in Image processing output

- References
- https://www.sciencedirect.com/topics/computer-science/sobel-filter#:~:text=A%20Sobel%20filter%2C%20in%20the,by%20calculating%20the%20gradient%20intensity.
- https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm#:~:text=The%20Sobel%20operator%20performs%20a,in%20an%20input%20grayscale%20image.
- https://www.nv5geospatialsoftware.com/docs/LaplacianFilters.html#:~:text=A%20Laplacian%20filter%20is%20an,an%20edge%20or%20continuous%20progression.
- https://www.sciencedirect.com/topics/engineering/laplacian-filter
- https://www.baeldung.com/cs/computer-vision-contours#:~:text=Contours%20play%20a%20key%20role,input%20image%20to%20process%20it.
- https://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm#:~:text=How%20It%20Works,with%20high%20first%20spatial%20derivatives.