Getting Started with OpenCV

```
#First, you need to install OpenCV. You can do this using pip:
!pip install opencv-python
```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.8.0.76)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python) (1.25.2)
```

1. Reading and Displaying an Image

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

# Read the image from file
image = cv2.imread('/content/image.jpeg')

# Display the image
cv2_imshow(image)
```

2. Convert an Image to Grayscale

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

# Read the image from file
image = cv2.imread('/content/image.jpeg')

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

# Display the grayscale image
cv2_imshow(gray_image)
```

3. Save an Image

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

# Read the image from file
image_path = '/content/image.jpeg'
image = cv2.imread(image_path)

# Check if the image was successfully read
if image is None:
    print(f"Error: Unable to read image from {image_path}")
else:
    # Convert to grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Display the grayscale image
    cv2_imshow(gray_image)

    # Save the grayscale image
    cv2.imwrite('/content/gray_image.jpg', gray_image)
```

4. Resize an Image

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

# Define the image path
image_path = '/content/image.jpeg'

# Read the image
image = cv2.imread(image_path)

# Check if the image was successfully read
if image is None:
    print(f"Error: Unable to read image from {image_path}")
else:
    # Resize the image
    resized_image = cv2.resize(image, (200, 200))

    # Display the resized image
    cv2_imshow(resized_image)

    # Optionally, save the resized image
    cv2.imwrite('/content/resized_image.jpg', resized_image)
```

5. Blur an Image

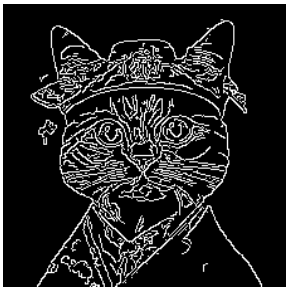
(15, 15): This is the size of the Gaussian kernel. The kernel size must be a positive odd number (e.g., 3, 5, 7, 9, etc.). In this case, a 15x15 kernel is used, which means the blur effect is calculated based on a 15x15 pixel area around each pixel in the image. A larger kernel size results in a stronger blur effect.

0: This is the standard deviation in the X direction (sigmaX). By setting it to 0, OpenCV will automatically calculate the appropriate value based on the kernel size. You can also specify a value for finer control over the blur effect.

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

# Define the image path
image_path = '/content/image.jpeg'

# Read the image
image = cv2.imread(image_path)

# Check if the image was successfully read
if image is None:
    print(f"Error: Unable to read image from {image_path}")
else:
    # Apply Gaussian blur
    blurred_image = cv2.GaussianBlur(image, (15, 15), 0)

    # Display the blurred image
    cv2_imshow(blurred_image)
```

6. Edge Detection

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

# Define the image path
image_path = '/content/image.jpeg'

# Read the image
image = cv2.imread(image_path)

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

# Apply Canny edge detection
edges = cv2.Canny(gray_image, 100, 200)

# Display the edges
cv2_imshow(edges)
```

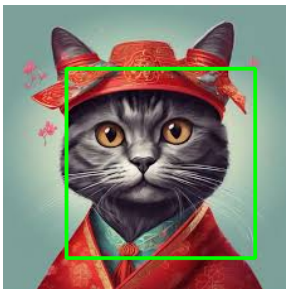


7. Draw a Line

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

# Define the image path
image_path = '/content/image.jpeg'

# Read the image
image = cv2.imread(image_path)

# Draw a line on the image
start_point = (50, 50)
end_point = (200, 200)
color = (255, 0, 0)  # BGR format
thickness = 2
cv2.line(image, start_point, end_point, color, thickness)

# Display the image with the line
cv2_imshow(image)
```

8. Draw a Rectangle

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

# Define the image path
image_path = '/content/image.jpeg'

# Read the image
image = cv2.imread(image_path)

# Draw a rectangle on the image
start_point = (50, 50)
end_point = (200, 200)
color = (0, 255, 0)  # BGR format
thickness = 2
cv2.rectangle(image, start_point, end_point, color, thickness)

# Display the image with the rectangle
cv2_imshow(image)
```

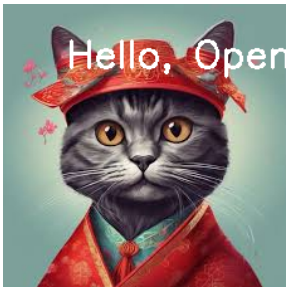


9. Draw a Circle

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

# Define the image path
image_path = '/content/image.jpeg'

# Read the image
image = cv2.imread(image_path)

# Draw a circle on the image
center_coordinates = (150, 150)
radius = 50
color = (0, 0, 255)  # BGR format
thickness = 2
cv2.circle(image, center_coordinates, radius, color, thickness)

# Display the image with the circle
cv2_imshow(image)
```

12. Write Text on an Image

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

# Define the image path
image_path = '/content/image.jpeg'

# Read the image
image = cv2.imread(image_path)

# Write text on the image
text = 'Hello, OpenCV!'
org = (50, 50)  # Bottom-left corner of the text
font = cv2.FONT_HERSHEY_SIMPLEX
font_scale = 1
color = (255, 255, 255)  # White color in BGR format
thickness = 2
cv2.putText(image, text, org, font, font_scale, color, thickness)

# Display the image with the text
cv2_imshow(image)
```



13. Split and Merge Channels

The line blank = np.zeros_like(b) creates a blank (black) image that has the same dimensions as the blue channel (b). Here's a breakdown of what it does:

np.zeros_like(b): This function call creates a new array of zeros with the same shape and type as the given array b. b is a 2D array representing one color channel (blue) of the image, so blank will be a 2D array of the same size filled with zeros.

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Define the image path
image_path = '/content/image.jpeg'

# Read the image
image = cv2.imread(image_path)

# Split the image into channels
b, g, r = cv2.split(image)

# Create blank channels
blank = np.zeros_like(b)

# Merge the channels with blank channels to visualize them in color
blue_channel = cv2.merge([b, blank, blank])
green_channel = cv2.merge([blank, g, blank])
red_channel = cv2.merge([blank, blank, r])

# Display each channel in its respective color
cv2_imshow(blue_channel)  # Blue channel
cv2_imshow(green_channel)  # Green channel
cv2_imshow(red_channel)  # Red channel
```

14. Convert an Image to HSV

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

# Define the image path
image_path = '/content/image.jpeg'  # Correct the file extension

# Read the image
image = cv2.imread(image_path)

# Check if the image was loaded successfully
if image is None:
    print("Error: Image not found or unable to load.")
else:
    # Convert BGR image to HSV
    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

15. Thresholding

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

# Define the image path
image_path = '/content/image.jpeg'

# Read the image
image = cv2.imread(image_path)

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

# Apply thresholding
ret, thresh = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)

# Display the thresholded image
cv2_imshow(thresh)
```