Basics of Computer Vision and OpenCV

1. What are Computer Vision and OpenCV?

Computer Vision

Computer Vision is a field of artificial intelligence that enables computers and machines to interpret and understand visual information from images and videos. It involves techniques for processing, analyzing, and extracting meaningful data from visual inputs, supporting applications such as facial recognition, object detection, and autonomous driving.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides infrastructure for developing computer vision applications, containing over 2,500 optimized algorithms. OpenCV supports multiple programming languages, including Python, C++, Java, and MATLAB.

---

2. Installation of OpenCV

Install OpenCV Using pip (Recommended for Python)

This method works across all operating systems:

pip install opencv-python

To install OpenCV with extra modules:

pip install opencv-contrib-python

Verify installation:

```
import cv2
print(cv2.__version__)
```

Install OpenCV from Source (For Advanced Users)

```
git clone https://github.com/opencv/opencv.git
git clone https://github.com/opencv/opencv_contrib.git
mkdir -p build && cd build
cmake -DOPENCV_EXTRA_MODULES_PATH=../opencv_contrib/modules ../opencv
make -j$(nproc)
sudo make install
```

Verify installation:

```
import cv2
print(cv2.__version__)
```

---

3. What are Images? How are Images Formed?

Definition

An image is a digital representation of visual information composed of pixels. Pixels are the smallest units of an image, each representing a specific color or intensity.

Formation of Images

Images are captured when light reflects off objects and is detected by sensors in devices like cameras. These sensors convert light into electrical signals, which are digitized into pixel values.

Image Storage on Computers

- **Grayscale Images**: Stored as 2D arrays, where each element represents the intensity of a pixel (0 to 255).
- **Color Images**: Stored as 3D arrays with three channels (Red, Green, Blue), where each pixel is represented by three intensity values.

Images in Python Using NumPy Arrays

```
from PIL import Image
import numpy as np

# Load image
image = Image.open('sample.png')

# Convert image to NumPy array
image_array = np.asarray(image)

# Display shape of array
print(image_array.shape)
```

This method helps in direct pixel manipulation for various image processing tasks.

---

4. How to Read and Write Images in OpenCV

Reading an Image

```
import cv2

# Load a color image
image_color = cv2.imread('path/to/image.jpg', cv2.IMREAD_COLOR)

# Load a grayscale image
image_gray = cv2.imread('path/to/image.jpg', cv2.IMREAD_GRAYSCALE)
```

Displaying an Image

```
cv2.imshow('Image', image_color)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Writing an Image

```
cv2.imwrite('path/to/saved_image.jpg', image_gray)
```

---

5. Grayscaling (Converting Color Images to Shades of Gray)

Method 1: Using cv2.cvtColor()

```
import cv2

# Load the color image
image = cv2.imread('path_to_image.jpg')

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

# Display grayscale image
cv2.imshow('Grayscale Image', gray_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Method 2: Reading Image Directly in Grayscale Mode

```
import cv2

# Load image in grayscale mode
gray_image = cv2.imread('path_to_image.jpg', 0)
```

Benefits of Grayscaling

- **Reduces complexity** by eliminating color information.
- **Enhances performance** of computer vision algorithms.

---

6. Understanding Color Spaces

Common Color Spaces

- **RGB (Red, Green, Blue)**: Standard model for digital images.
- **HSV (Hue, Saturation, Value)**: Separates color information (hue) from intensity (value), useful for color-based segmentation.
- **Grayscale**: Represents only intensity without color.

Converting Between Color Spaces

```
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

---

7. Histogram Representation of Images

Histograms help visualize pixel intensity distribution, useful for image contrast and brightness analysis.

Compute Histogram in OpenCV

histogram = cv2.calcHist([image_gray], [0], None, [256], [0, 256])

---

8. Creating and Drawing on Images in OpenCV

1. Creating a Blank Image

```
import numpy as np
import cv2

# Create a black image
image = np.ones((400, 400, 3), dtype="uint8")*255
```

2. Drawing a Square (Rectangle)

#image = cv2.rectangle(image, start_point, end_point, color, thickness)

cv2.rectangle(image, (50, 50), (150, 150), (255, 0, 0), thickness=2)

3. Drawing a Circle

#cv2.circle(image, center_coordinates, radius, color, thickness)

cv2.circle(image, (200, 200), 50, (0, 255, 0), thickness=2)

4. Drawing a Polygon

#cv2.polylines(image, [pts], isClosed, color, thickness)

```
points = np.array([[300, 300], [350, 250], [400, 300], [350, 350]], np.int32)
points = points.reshape((-1, 1, 2))
cv2.polylines(image, [points], isClosed=True, color=(0, 0, 255), thickness=2)
```

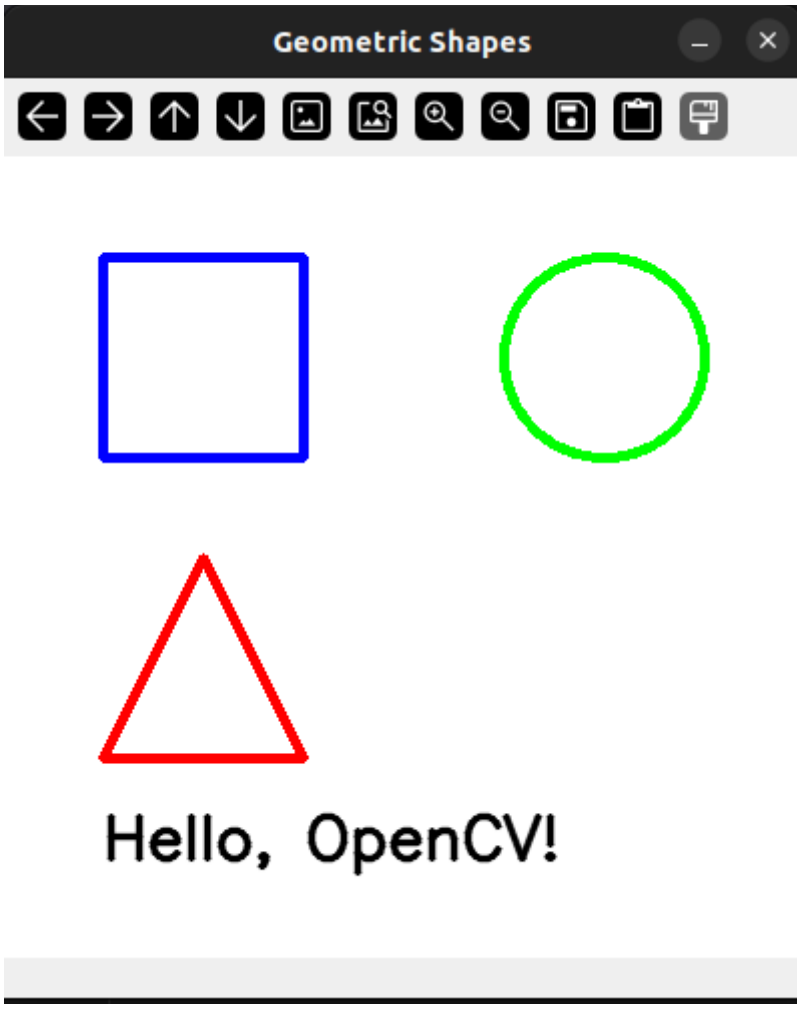
5. Adding Text

#image = cv2.putText(image,'OpenCV',org,font,fontScale,color,thickness, cv2.LINE_AA)

cv2.putText(image, 'OpenCV', (50, 350), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), thickness=2)

6. Displaying the Image

```
cv2.imshow('Image with Shapes', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



This sequence creates a white image, draws a blue square, a green circle, a red polygon, and adds white text 'OpenCV'.
