OpenCV Fundamentals – Week 1 – Assignment

1. Introduction to OpenCV
Overview & History:

OpenCV (Open Source Computer Vision Library) is an open-source library originally developed by Intel in 1999. It focuses on real-time computer vision, image processing, and machine learning applications. It supports C++, Python, Java, and works on Windows, Linux, macOS, Android, and iOS.

Installation (Python)

```
pip install opencv-python opencv-contrib-python
```

Basic Setup

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

2. Core Module (cv2.core)

Uses NumPy arrays to store image data.

Supports basic operations: addition, subtraction, bitwise operations, type conversions.

Example:

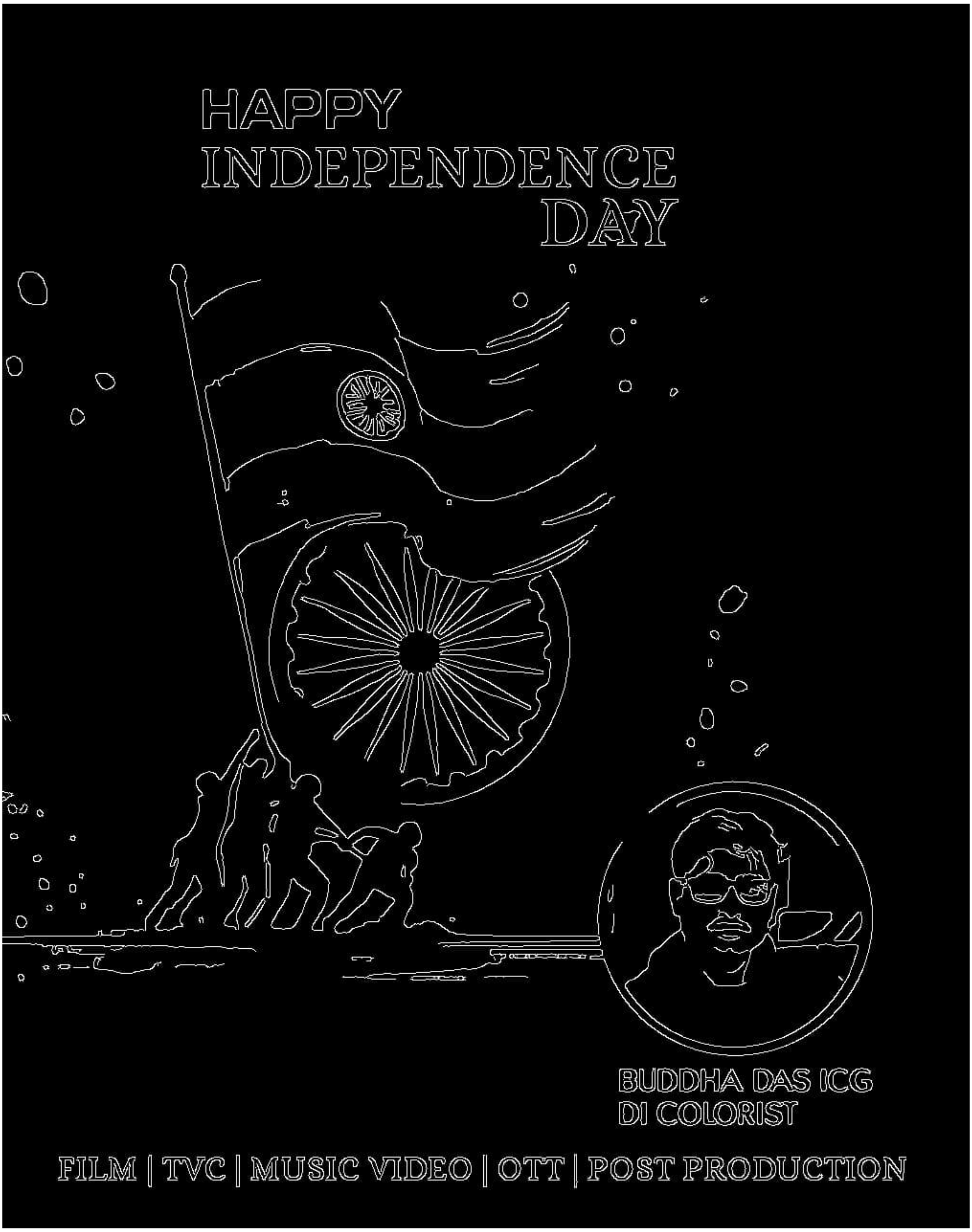
```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Create a black image with 3 channels (for BGR color),
# 300x300 pixels, and an 8-bit unsigned integer data type.
# The `np.zeros` function initializes all pixel values to 0 (black).
img = np.zeros((300, 300, 3), dtype=np.uint8)
```

```
# Change the color of the entire image to green.
# In OpenCV, colors are represented in BGR (Blue, Green, Red) format.
# (0, 255, 0) corresponds to a full green color.
img[:] = (0, 255, 0)

# Display the image in a window named "Core Module Demo".
cv2_imshow(img)

# cv2.waitKey(0) is not needed with cv2_imshow in Colab
# cv2.destroyAllWindows() is also not needed
```



3. Image Processing (cv2.imgproc)

Transformations: resize, rotate, flip.

Filtering: blur, Gaussian blur, median blur.

Geometric operations: affine and perspective transforms.

Histograms: visualization of pixel intensity distribution.

Example:

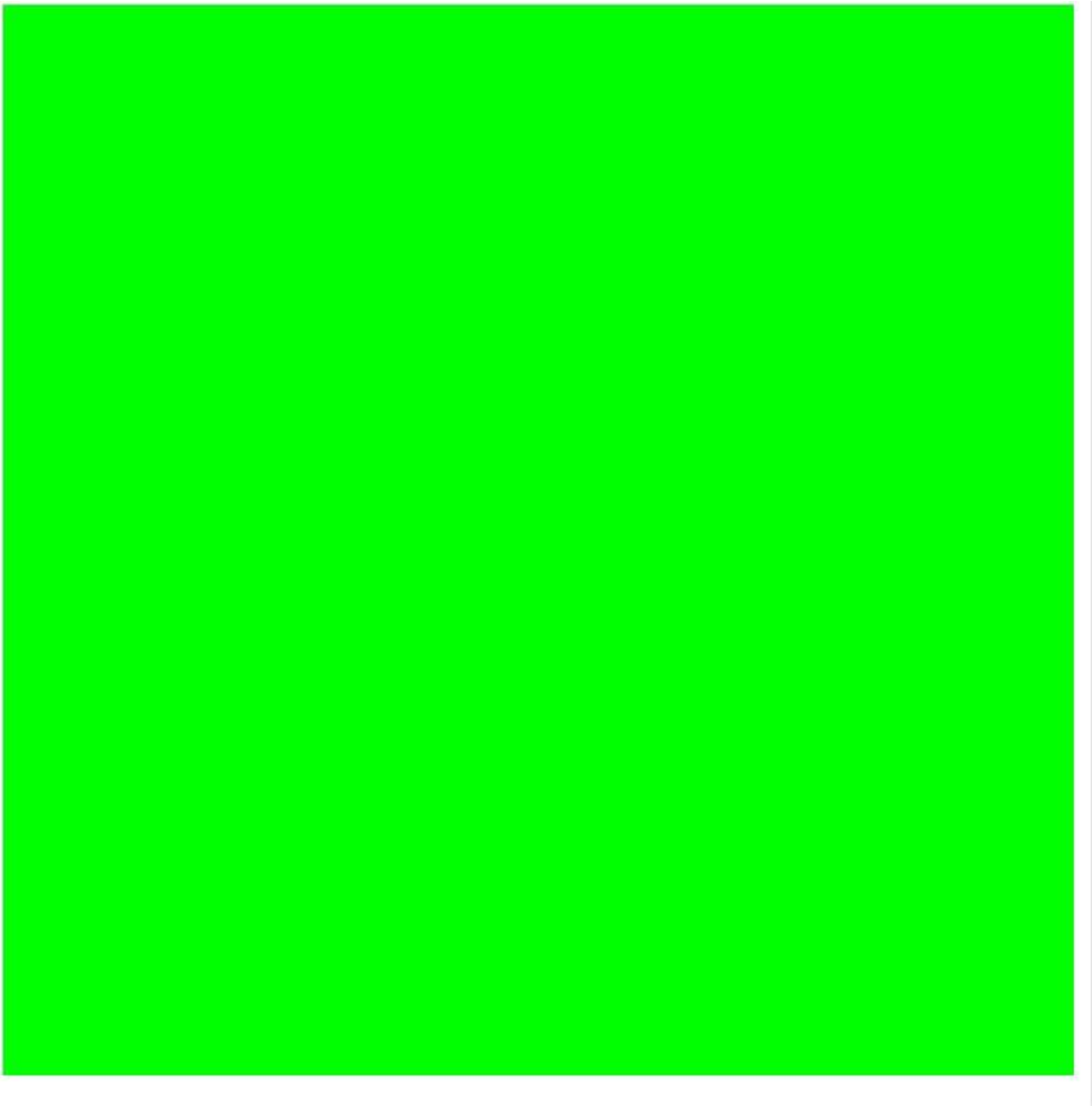
```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

OUTPUT:

```
IMG-20250815-WA0013.jpg
IMG-20250815-WA0013.jpg(image/jpeg) - 90996 bytes, last modified: 8/15/2025 - 100%
done
Saving IMG-20250815-WA0013.jpg to IMG-20250815-WA0013.jpg
User uploaded file "IMG-20250815-WA0013.jpg" with length 90996
```



4. Application Utilities (highgui, imgcodecs, videoio)

highgui: GUI display functions (cv2.imshow, cv2.waitKey, cv2.destroyAllWindows).
imgcodecs: Read and write images (cv2.imread, cv2.imwrite).
videoio: Read/write video files and access webcams.

Example:

```
# Read and save an image
img = cv2.imread("sample.jpg")
cv2.imwrite("saved_image.jpg", img)
```

5. Camera Calibration & 3D Reconstruction (cv2.calib3d)

Used to correct image distortion from lenses.
Used for stereo vision (depth perception).

Example: Undistortion (requires camera matrix & distortion coefficients)

```
# This is just the method; actual calibration requires chessboard images
dst = cv2.undistort(img, cameraMatrix, distCoeffs, None)
```

6. Object Detection (cv2.objdetect)

Detect objects like faces, eyes, etc., using Haar cascades.

Example: Face Detection

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
img = cv2.imread("faces.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.1, 4)

for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

cv2.imshow("Faces", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

7. 2D Features Framework (cv2.feature2d)

Detects and matches key points in images (e.g., SIFT, ORB).

Example: ORB Feature Detection

```
img = cv2.imread("sample.jpg", 0)
orb = cv2.ORB_create()
kp, des = orb.detectAndCompute(img, None)
out_img = cv2.drawKeypoints(img, kp, None, color=(0,255,0))

cv2.imshow("ORB Features", out_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8. Deep Neural Networks (cv2.dnn)

Load and run pre-trained deep learning models.

Example: Load a Caffe model

```
net = cv2.dnn.readNetFromCaffe("deploy.prototxt", "model.caffemodel")
```

9. Graph API (cv2.gapi)

Pipeline-based processing for performance optimization.

Mostly used in complex applications.

10. Other Tutorials

ml: Machine learning with OpenCV.

photo: Denoising, image restoration.

stitching: Panorama creation.

video: Motion detection, tracking.

11. Theory of Image Processing

Pixels: smallest unit of an image.

Color spaces: BGR, RGB, HSV, Gray.

Convolution: kernel sliding for filtering.

Edge detection: Canny, Sobel, Laplacian.

Fourier Transform: frequency analysis.

12. Applications of OpenCV

1. Face recognition – security systems.

2. Autonomous vehicles – lane detection.

3. Medical imaging – tumor detection.

4. Robotics – object tracking.

5. Augmented reality – marker detection.

6. Industrial automation – defect detection.

7. Surveillance – motion tracking.

8. Agriculture – plant disease detection.

9. Sports analytics – player tracking.

10. Document scanning – perspective correction.

12. Linux vs Windows OpenCV Window Differences

On Windows, cv2.imshow() can run without a main loop; event handling is managed internally.

On Linux, GUI windows are more dependent on cv2.waitKey() for event processing.

Multi-window handling is more stable on Windows; Linux may require threading workarounds.