Digital Image Processing

Lab 02

---

01. Convert BGR image to Grayscale Without using cvtColor Function

In [1]:

```
import cv2
```

In [2]:

```
path = "C:\\Users\\hp\\Google Drive\\Fiverr Work\\2022\\33. Computer Vision Course\\pict
ures"
```

In [3]:

```
img = cv2.imread(path+"\\coloredChips.png")  # took path and name of image as an argument
```

1.1 Split the channles

In [4]:

```
blueChannel = img[:,:,0]
greenChannel = img[:,:,1]
redChannel = img[:,:,2]
```

1.2 Convert it to Grayscale

In [9]:

```
grayImg = (0.299 * redChannel + 0.587 * greenChannel + 0.114 * blueChannel)/255
```

Why we need to multiply these values with each cahnnel?

When converting a BGR image to grayscale, we need to take into account the different contributions of each color channel to the perceived brightness of a pixel.

The human eye is more sensitive to green light than red or blue, so we should assign a higher weight to the green channel in the grayscale conversion formula. The coefficients 0.299, 0.587, and 0.114 that we use in the formula are based on the relative luminance of each color channel, taking into account the sensitivity of the human eye to each color.

Therefore, we multiply each color channel by its corresponding coefficient to compute the grayscale value for each pixel. This ensures that the resulting grayscale image accurately represents the relative brightness of the original color image, while also taking into account the human eye's sensitivity to different colors.

Why these specific values? where they come from?

The values 0.299, 0.587, and 0.114 that are commonly used in the grayscale conversion formula for BGR images are based on the relative luminance of each color channel, as well as the sensitivity of the human eye to different

colors.

The coefficients were originally derived from the CIE (Commission Internationale de l'Eclairage) color space, which defines the standard observer model for the human eye. The CIE model includes a set of color matching functions that describe the spectral sensitivity of the human eye to different colors.

The coefficients used in the grayscale conversion formula are based on the CIE color matching functions, and are designed to approximate the perceived brightness of a color image for a standard observer under typical viewing conditions. Specifically, the coefficients are designed to take into account the fact that the human eye is most sensitive to green light, followed by red and then blue light.

The specific values of 0.299, 0.587, and 0.114 were chosen to provide a good balance between accuracy and simplicity, and have become widely adopted as a standard for grayscale conversion in BGR images.

1.3 Display Grayscale Image

In [10]:

```
cv2.imshow("Original Image", img)
cv2.imshow("Grayscale Image", grayImg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Full Code

In [17]:

```
import cv2

path = "C:\\Users\\hp\\Google Drive\\Fiverr Work\\2022\\33. Computer Vision Course\\pict
ures"

img = cv2.imread(path+"\\coloredChips.png")  # took path and name of image as an argument

blueChannel = img[:,:,0]
greenChannel = img[:,:,1]
redChannel = img[:,:,2]

grayImg = (0.299 * redChannel + 0.587 * greenChannel + 0.114 * blueChannel)/255

cv2.imshow("Original Image", img)
cv2.imshow("Grayscale Image", grayImg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

---

02. Use Mobile Phone Camera as Webcam

- **Download app iriun Webcam app from play store into your mobile** https://play.google.com/store/apps/details?id=com.jacksoftw.webcam&hl=en&gl=US
- **Also download software iriun for windows into your laptop** https://iriun.com/

Full Code

In [ ]:

```
import cv2

cam = cv2.VideoCapture(1)
```

```
while True:
    Success, Frame = cam.read()

    cv2.imshow("Mobile Cam", Frame)
    k = cv2.waitKey(1)

    if k == ord("q"):
        break

cam.release()
cv2.destroyAllWindows()
```

---

03. Screen Recorder Using OpenCV

In [11]:

```
import cv2
import pyautogui
import numpy as np
```

3.1 Get Screen Resolution

In [12]:

```
# create resolution
screenResolution = pyautogui.size()      # return screen width and height
```

3.2 Get the file Name and Path where you want to store recording

In [13]:

```
# file name in which we want to store recording
fileName = input("Enter file name and path: ")
```

3.3 Define fourcc

cv2.VideoWriter_fourcc is a function in the OpenCV library for Python that is used to create a four-character code (fourcc) that specifies the video codec to be used when writing a video file using cv2.VideoWriter.

The fourcc code is a 32-bit integer that is used to identify the video codec used to encode the video frames. Different video codecs have different fourcc codes, and the fourcc code can be used to specify the codec when opening a video file for writing.

The function takes four arguments, which are the four characters that make up the fourcc code. These characters can be any ASCII characters, and they are combined into a 32-bit integer using bitwise operations.

In [14]:

```
# Now fix the frame rate
fps = 30

fourcc = cv2.VideoWriter_fourcc(*'XVID')
output = cv2.VideoWriter(fileName, fourcc, fps, screenResolution)
```

3.4 Creating Recording Window

In [15]:

```
# create recording module
cv2.namedWindow("Live Recording", cv2.WINDOW_NORMAL)
```

```
cv2.resizeWindow("Live Recording", (640, 480))
```

3.5 Start Recording

In [16]:

```
while True:
    img = pyautogui.screenshot()
    f = np.array(img)

    f = cv2.cvtColor(f, cv2.COLOR_BGR2RGB)

    output.write(f)
    cv2.imshow("Live Recording", f)

    k = cv2.waitKey(1)

    if k == ord("q"):
        break

output.release()
cv2.destroyAllWindows()
```

Full Code

In [ ]:

```
# Screen Recoder

import cv2
import pyautogui
import numpy as np

# create resolution
screenResolution = pyautogui.size()     # return screen width and height

# file name in which we want to store recording
fileName = input("Enter file name and path: ")

# Now fix the frame rate
fps = 30

fourcc = cv2.VideoWriter_fourcc(*'XVID')
output = cv2.VideoWriter(fileName, fourcc, fps, screenResolution)

# create recording module
cv2.namedWindow("Live Recording", cv2.WINDOW_NORMAL)
cv2.resizeWindow("Live Recording", (640, 480))

while True:
    img = pyautogui.screenshot()
    f = np.array(img)

    f = cv2.cvtColor(f, cv2.COLOR_BGR2RGB)

    output.write(f)
    cv2.imshow("Live Recording", f)

    k = cv2.waitKey(1)

    if k == ord("q"):
        break

output.release()
cv2.destroyAllWindows()
```

04. Extracting Frames from Video or Webcam

Full Code

In [20]:

```
# extracting the frames from video

import cv2

cam = cv2.VideoCapture("traffic.avi")
Success, frame = cam.read()
count = 0

while True:
    if Success:
        cv2.imwrite(f"frames\\imgn{count}.jpg", frame)

        # setting the frame speed
        cam.set(cv2.CAP_PROP_POS_MSEC, (count**100))

        Success, frame = cam.read()

        cv2.imshow("frame extraction", frame)

        count +=1
        k = cv2.waitKey(1)
        if k == ord('q'):
            break
            cv2.destroyAllWindows()

cam.release()
cv2.destroyAllWindows()
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
Cell In [20], line 18
     14 cam.set(cv2.CAP_PROP_POS_MSEC, (count**100))
     16 Success, frame = cam.read()
---> 18 cv2.imshow("frame extraction", frame)
     20 count +=1
     21 k = cv2.waitKey(1)

error: OpenCV(4.6.0) D:\a\opencv-python\opencv-python\opencv\modules\highgui\src\window.c
pp:967: error: (-215:Assertion failed) size.width>0 && size.height>0 in function 'cv::ims
how'
```

In [21]:

```
cam.release()
cv2.destroyAllWindows()
```

4.1 Save on pressing "S" button

In [22]:

```
import cv2

cam = cv2.VideoCapture(0)

count = 0

while True:

    Success, frame = cam.read()

    if Success:
```

```
        # # setting the frame speed
        # cam.set(cv2.CAP_PROP_POS_MSEC, (count**100))

        cv2.imshow("frame extraction", frame)

        count +=1
        k = cv2.waitKey(1)

        if k == ord('q'):
            cv2.destroyAllWindows()
            break
        elif k == ord("s"):
            cv2.imwrite(f"frames\\imgn{count}.jpg", frame)


cam.release()
cv2.destroyAllWindows()
```

In [ ]:

```
```