Image Processing

کتابخانه OpenCV :

برای کار کردن با دیتاست هایی که به صورت تصویر هستند از کتابخانه OpenCV استفاده می کنیم.

❖ درک سیستم از تصویر :

سیستم ، تصویر را مجموعه ای از پیکسل ها در نظر می گیرد. برای مثال اگر بگوییم اندازه تصویری ۷۲۰ در ۱۰۸۰ است بدین معنی است که ۷۲۰ پیکسل در راستای y و ۱۰۸۰ پیکسل در راستای x دارد.

هر پیکسل تشکیل شده از سه رنگ RGB است که R به معنی Red ، G به معنی Green و B به معنی Blue است. هر رنگ دارای بازه عددی 0 تا 255 است که 0 به معنی مشکی است و 255 به معنی همان رنگ به صورت خالص است. به عبارت دیگر هر پیکسل تشکیل شده از سه مقدار یا سه عدد است که در نهایت یک رنگ تولید خواهد کرد.

به هر کدام از این رنگ ها کانال گفته می شود. در نهایت به تصویری که شامل رنگ های RGB باشد ، سه کاناله می گویند.

نکته ! تصاویر می توانند تک کاناله باشند. یعنی دیگر کانال های رنگی ندارد و تصویر شما به صورت رنگی نخواهد بود. اصطلاحا به این تصاویر Gray می گویند.

نکته ! تصاویر تک کاناله حجم پردازش را کم می کنند.

نکته ! اگر تصویر فقط سیاه و سفید باشد یعنی مقدار پیکسل ها فقط 0 یا 255 باشد به آن تصویر باینری می گویند.

نکته ! مبدا دستگاه مختصاتی که بر روی تصویر قرار می گیرد ، بالای تصویر سمت چپ آن خواهد بود.

اگر تصاویر را با استفاده از کتابخانه OpenCV بخوانیم به عنوان خروجی یک آرایه بر می گرداند.

اگر این آرایه سه بعدی باشد بدین معنی است که تصویر سه کاناله رنگی است.

نکته ! توجه داشته باشید که در OpenCV کانال ها برعکس هستند یعنی به جای RGB ، BGR هستند. برای اینکه این مشکل را بتوانیم حل کنیم باید از دستور خاصی استفاده کنیم.

❖ **کار کردن با تصاویر با استفاده از OpenCV :**

می خواهیم یک تصویر را با استفاده از کتابخانه OpenCV بخوانیم. داریم :

```
import cv2

img = cv2.imread("photo.jpg")
print(img.shape)
```

اگر بخواهیم فقط قسمتی از تصویر را بخوانیم. داریم :

```
x = img[100:300, 300:500]
cv2.imshow("img", x)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

برای تبدیل تصاویر BGR به RGB داریم :

```
rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
cv2.imshow("rgb_img", rgb_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

برای تبدیل تصاویر سه کاناله یا BGR به تک کاناله داریم :

```
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow("gray_img", gray_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

اگر فقط یکی از لایه های RGB را بخواهیم. داریم :

```
cv2.imshow("green_img", img[:, :, 2])
cv2.waitKey(0)
cv2.destroyAllWindows()
```

برای ذخیره عکس داریم :

```
cv2.imwrite("output.png", img)
```

❖ **کار کردن با ویدیو با استفاده از OpenCV :**

می خواهیم یک ویدیو را با استفاده از کتابخانه OpenCV بخوانیم. داریم :

```
video = cv2.VideoCapture("video.mp4")

while True:
    r, f = video.read()

    if f is None: break

    cv2.imshow("frame", f)

    if cv2.waitKey(34) == ord("e"): break
```

نکته ! پارامتر ورودی دستور VideoCapture() می تواند آدرس ویدیویی در سیستم باشد ، همچنین می تواند مقدار 0 باشد که در این صورت به وبکم سیستم دسترسی پیدا می کند ، مقادیر 1, 2, ... برای استفاده از دوربین خارجی می باشد.

❖ **رسم اشکال مختلف بر روی تصویر با استفاده از OpenCV :**

اگر بخواهیم خط ساده بر روی تصویر رسم کنیم ، داریم :

```
img = cv2.imread("photo.jpg")

cv2.line(img, (15, 30), (300, 80), (0, 255, 0), 5)

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

اگر بخواهیم کادر مستطیل شکل بر روی تصویر رسم کنیم ، داریم :

```
img = cv2.imread("photo.jpg")

cv2.rectangle(img, (15, 30), (300, 80), (0, 255, 0), 5)

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

اگر بخواهیم یک دایره بر روی تصویر رسم کنیم ، داریم :

```
img = cv2.imread("photo.jpg")

cv2.circle(img, (300, 300), 80, (0, 255, 0), 5)

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

اگر بخواهیم متنی بر روی تصویر بنویسیم ، داریم :

```
img = cv2.imread("photo.jpg")

cv2.putText(img, "Image Processing", (150, 150), cv2.FONT_HERSHEY_SIMPLEX,
            1.15, (0, 0, 255), 5)

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

پروژه :

می خواهیم با استفاده از کتابخانه OpenCV و مدل های یادگیری ماشینی که تا کنون یاد گرفته ایم ، برنامه ای بنویسیم که عکس دریافت کند و تشخیص دهد که روز است یا شب. داریم :

```
import glob
import numpy as np
import cv2
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from joblib import dump

def load_data():
    feature_vectors = []
    labels = []

    for address in glob.glob("images\\*\\*\\*"):
        img = cv2.imread(address)
        img = cv2.resize(img, (32, 32))
        img = img/255.0
        img = img.flatten()

        feature_vectors.append(img)

        label = address.split("\\")[2]
        labels.append(label)

    feature_vectors = np.array(feature_vectors)

    X_train, X_test, y_train, y_test = train_test_split(feature_vectors,
                                                        labels, test_size=0.2,
                                                        random_state=42)

    return X_train, X_test, y_train, y_test
```

```
X_train, X_test, y_train, y_test = load_data()

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)

dump(knn, "day_night_classifier.z")

y_pred = knn.predict(X_test)
print(classification_report(y_test, y_pred))
```

Written by Hossein Kariznoei

پس از ذخیره مدل برای استفاده از مدل داریم :

```
import numpy as np
import glob
import cv2
from joblib import load

knn = load("day_night_classifier.z")

for address in glob.glob("test_model\\*"):
    img = cv2.imread(address)
    r_img = cv2.resize(img, (32, 32))
    r_img = r_img/255.0
    r_img = r_img.flatten()
    r_img = np.array([r_img])

    label = knn.predict(r_img)[0]

    cv2.putText(img, label, (20, 20),
                cv2.FONT_HERSHEY_SIMPLEX, 1.15,
                (0, 0, 255), 5)

    cv2.imshow("img", img)
    cv2.waitKey(0)

cv2.destroyAllWindows()
```