Color tracking :

با استفاده از متد inRange می توانیم بخش هایی از یک ویدئو را نیز track کنیم. به کد زیر دقت کنید:

```
cap = cv2.VideoCapture("videos/blue-track.mp4")

while True:
    ret, frame = cap.read()
    if not ret:
        break
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    lower_blue = np.array([100,50,50])
    upper_blue = np.array([130,255,255])

    mask = cv2.inRange(hsv, lower_blue, upper_blue)
        contours,    _    =    cv2.findContours(mask,    cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    centers, radius = cv2.minEnclosingCircle(contours[0])
    centers = int(centers[0]), int(centers[1])
    radius = int(radius)
    cv2.circle(frame, centers, radius, (0,0,255), 2)
    cv2.imshow('image',frame)
    cv2.imshow('mask',mask)
    cv2.imshow('res',res)

    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break

cv2.destroyAllWindows()
cap.release()
```

این کد با استفاده از OpenCV یک ویدیو را فریم‌به‌فریم تحلیل می‌کند و ناحیه‌ای با رنگ آبی را شناسایی کرده و سپس یک دایره که آن ناحیه را محاط کرده رسم می‌کند.

ابتدا فایل ویدیویی `"blue-track.mp4"` از مسیر `videos/` باز می‌شود و شیء `cap` به‌عنوان یک **کنترل‌گر ویدیو** استفاده می‌شود تا فریم‌ها را یکی‌یکی بخواند.

در ادامه حلقه‌ای بی‌نهایت که فریم‌ها را تا پایان ویدیو یا فشردن کلید خاصی ادامه می‌دهد.

- `ret`: مقدار بولی، اگر `True` باشد یعنی فریم به‌درستی خوانده شده.
- `frame`: تصویر فعلی ویدیو.
- اگر فریمی وجود نداشته باشد (مثلاً ویدیو تمام شود)، حلقه متوقف می‌شود.
- فریم را از فضای رنگی `BGR` به `HSV` تبدیل می‌کند.
- فضای HSV برای تشخیص رنگ‌ها دقیق‌تر است.

```
    lower_blue = np.array([100,50,50])
    upper_blue = np.array([130,255,255])

```

- این دو آرایه، محدوده پایین و بالای رنگ آبی را در فضای HSV مشخص می‌کنند.
- مناسب برای تشخیص رنگ‌های بین آبی تیره تا آبی روشن.

```
mask = cv2.inRange(hsv, lower_blue, upper_blue)
```

- ماسکی ایجاد می‌کند که فقط نواحی آبی تصویر در آن سفید هستند (255)، بقیه سیاه.(0)
- نوع خروجی: تصویر سیاه‌وسفید با همان اندازه تصویر ورودی.

این خط از کد در پردازش تصویر با استفاده از کتابخانه OpenCV ، برای **شناسایی نواحی خاصی از تصویر بر اساس رنگ** به کار می‌رود. به طور خاص، در این مثال هدف تشخیص رنگ آبی است.

قبل از این خط، تصویر اصلی که معمولاً در فضای رنگی BGR است (به فضای رنگی HSV تبدیل شده است. فضای HSV که مخفف Hue ، Saturation، و Value است (به طور خاص برای پردازش رنگ‌ها مناسب‌تر از BGR است؛ زیرا در HSV ، رنگ (Hue) به صورت جداگانه از روشنایی و شدت تعریف شده و این باعث می‌شود بتوان رنگ‌ها را بهتر تشخیص داد، مخصوصاً در شرایط نوری مختلف.

در این خط از کد، تابع `cv2.inRange()` بررسی می‌کند که کدام پیکسل‌های تصویر HSV دارای مقادیر در بازه‌ای مشخص هستند. بازه‌ی مورد نظر بین دو مقدار تعریف شده است:

```
lower_blue = np.array([100, 50, 50])
upper_blue = np.array([130, 255, 255])
```

این دو مقدار به ترتیب حد پایین و بالای رنگ آبی در فضای HSV را مشخص می‌کنند. یعنی هر پیکسلی که در تصویر HSV دارای:

- Hueبین 100 تا 130
- Saturationبین 50 تا 255
- Valueبین 50 تا 255

باشد، به عنوان یک پیکسل "آبی" شناسایی می‌شود.

تابع `cv2.inRange()`یک تصویر جدید به نام `mask` تولید می‌کند. این تصویر به صورت سیاه و سفید تک‌کاناله یا grayscale است:

- هر پیکسلی که در محدوده‌ی تعریف‌شده قرار داشته باشد، مقدار 255 (سفید) خواهد داشت.
- سایر پیکسل‌ها مقدار 0 (سیاه) خواهند داشت.

```
contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

- `findContours` لبه‌ها یا مرزهای شکل‌های سفید روی ماسک (نواحی آبی در تصویر) را پیدا می‌کند.
- `cv2.RETR_TREE`: ساختار سلسله‌مراتبی کانتورها را بازیابی می‌کند.
- `cv2.CHAIN_APPROX_SIMPLE`: نقاط غیرضروری را حذف می‌کند تا حافظه کمتری مصرف شود.
- خروجی :`contours`لیستی از کانتورها، که هر کدام یک آرایه از نقاط مرزی یک شیء است.

```
centers, radius = cv2.minEnclosingCircle(contours[0])
```

این خط از کد برای **محاسبه کوچک‌ترین دایره‌ای** است که بتواند **یک کانتور مشخص** را به طور کامل **درون خود جای دهد** .این دایره، به آنچه در ریاضیات به عنوان *دایره محاطی (minimum enclosing circle)* شناخته می‌شود، اشاره دارد.

1. `contours[0]`

- `contours`یک **لیست از کانتورها** است که از تابع `cv2.findContours()`به دست آمده است.
- هر عضو این لیست، مجموعه‌ای از نقاط مرزی یک ناحیه (شکل) خاص در تصویر است.
- `contours[0]`به **اولین کانتور** در لیست اشاره می‌کند (مثلاً بزرگ‌ترین یا اولین ناحیه‌ی شناسایی شده).

2. `cv2.minEnclosingCircle(...)`

- این تابع از OpenCV استفاده می‌شود برای:

- پیدا کردن **مرکز** و **شعاع** کوچک‌ترین دایره‌ای که کل نقاط کانتور را **در بر بگیرد**.
- این دایره ممکن است دقیقاً منطبق بر شکل کانتور نباشد، ولی تضمین می‌شود که تمام نقاط آن کانتور داخل یا روی محیط این دایره قرار دارند.

3.خروجی تابع:

`centers`: مختصات مرکز دایره به صورت یک **عدد اعشاری** یاfloat ، مانند `(245.6, 118.3)`.

`radius`: شعاع دایره (همچنین عددfloat ، مثل `34.2`