

Video Processing

OpenCV היא ספריית קוד פתוח פופולרית לעיבוד תמונה ווידאו.

היא מאפשרת לנו "ללמד את המחשב לראות" – לזהות עצמים, לעקוב אחר תנועה, לבצע פילטרים, לנתח צבעים ועוד.

הספרייה פותחה במקור על ידי אינטל, והיא כתובה ב-`C++`, אך היום נפוצה מאוד בשפת `Python` – במיוחד בתחומים של ראייה ממוחשבת, רובוטיקה, בינה מלאכותית ומערכות חכמות.

בעזרת `OpenCV` אנחנו יכולים לקחת תמונה או סרטון ולהפוך אותם לנתונים: לדוגמה – לזהות כדור מתגלגל, לעקוב אחרי בלון בשמיים, למדוד צבעים, לזהות פנים, ואפילו לבצע חיזוי תנועה.

עבודה בסיסית עם הספרייה:

```
cap = cv2.VideoCapture("ball.mp4")
```

cap הוא כמו "שלט" שמאפשר לנו לגשת לפריימים (תמונות בודדות) מתוך הסרט.

קורא את הפריים הראשון מהסרטון.

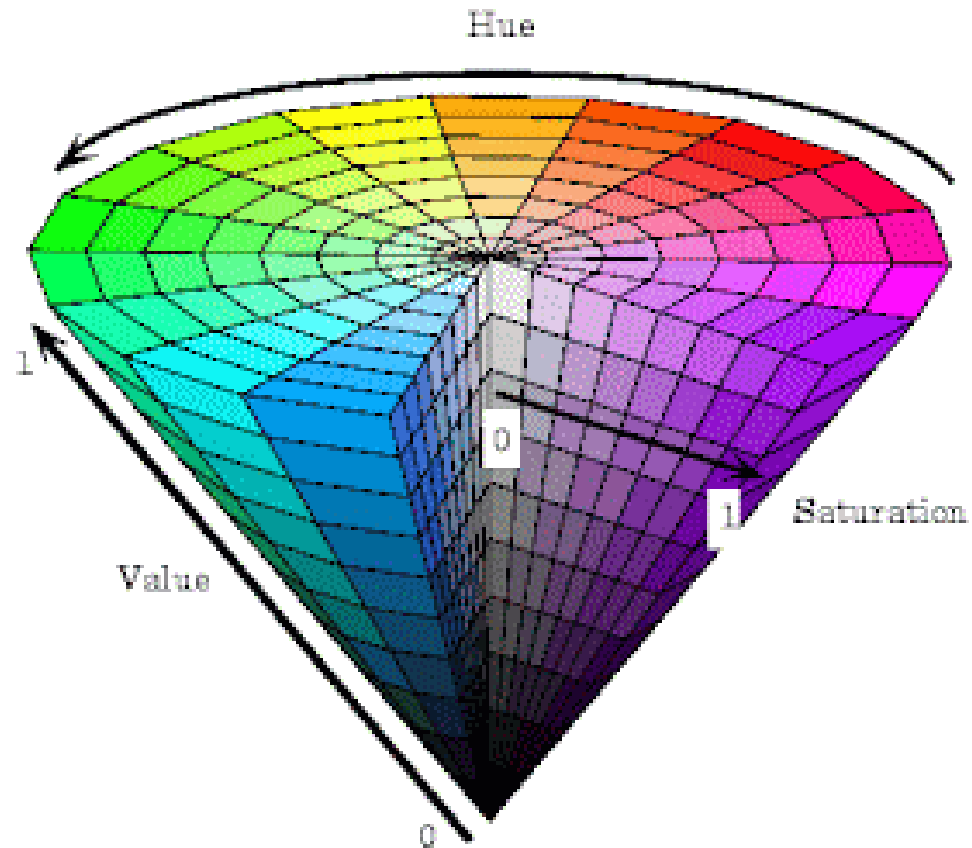
frame הוא התמונה שנקלטה

ret הוא ערך בוליאני (True/False) שמציין האם

הקריאה הצליחה

```
ret, frame = cap.read() # קורא את הפריים הראשון  
cap.release()
```

HSV הוא מודל צבע שמפרק כל צבע לשלושה רכיבים



H - Hue גוון: הצבע עצמו (למשל אדום, ירוק,

כחול...) - נמדד בזווית (0-179)

S - Saturation רוויה: כמה הצבע "עשיר" או

"אפור" - 0 זה אפור, 255 זה צבע חזק

V - Value בהירות: כמה הצבע בהיר - 0 זה שחור,

255 זה בהיר

שיטות צביעה נוספות ומתי משתמשים בהן

מתי נעדיף אותה?	שיטה
לזיהוי צבעים בתנאי תאורה משתנים	HSV
להצגה, שמירה, ציור, עיבוד בסיסי	RGB
לזיהוי קצוות, פילטרים, חישובים מהירים	GRAY
כשצריך התאמה מדויקת לצבעי ראייה אנושית	LAB

Grayscale



HSV



LAB



YCrCb

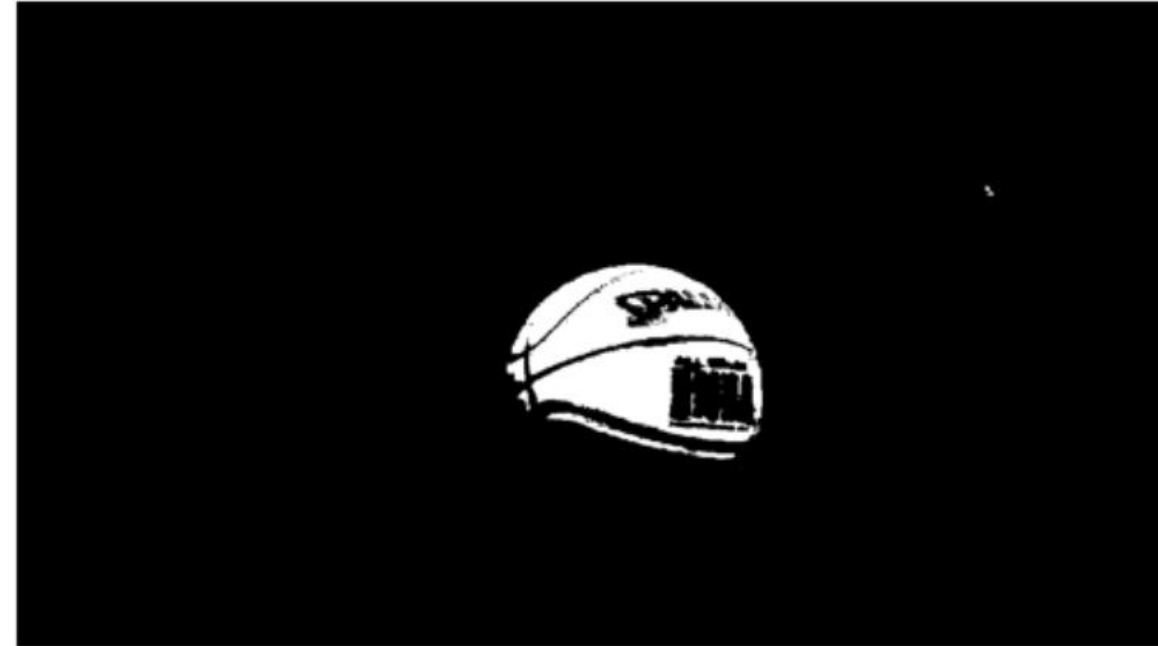


מסיכה היא תמונה בשחור-לבן שמסמנת אילו חלקים בתמונה מעניינים אותנו:

255 (לבן) = אזור פעיל / רלוונטי (שחור) = אזור לא רלוונטי / מוסתר

הפונקציה מקבלת frame וגבולות מינ' ומקס' של גוון, רוויה ובהירות

```
mask = cv2.inRange(hsv_frame, lower, upper)
```



מה זה קונטור? (Contour)

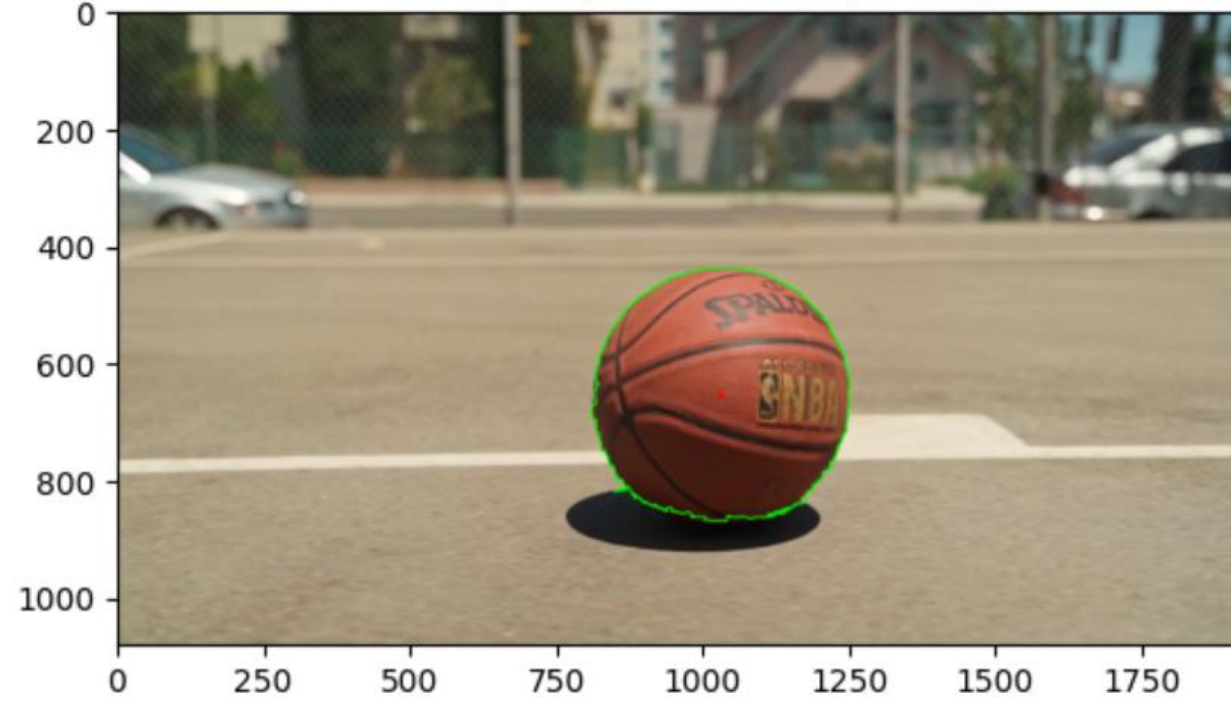
קונטור הוא קו שמקיף גבולות של צורה או אובייקט בתמונה.

ב- OpenCV, קונטור הוא בעצם **רשימת נקודות** שמשרטטות את קווי המתאר של אזור כלשהו.

למה זה שימושי?

- למציאת **אובייקטים** בתמונה
- למדידת **שטח, צורה, מרכז מסה**
- למעקב אחרי אובייקט בתנועה יחד עם Kalman למשל

Center(Red) and contour (Green)



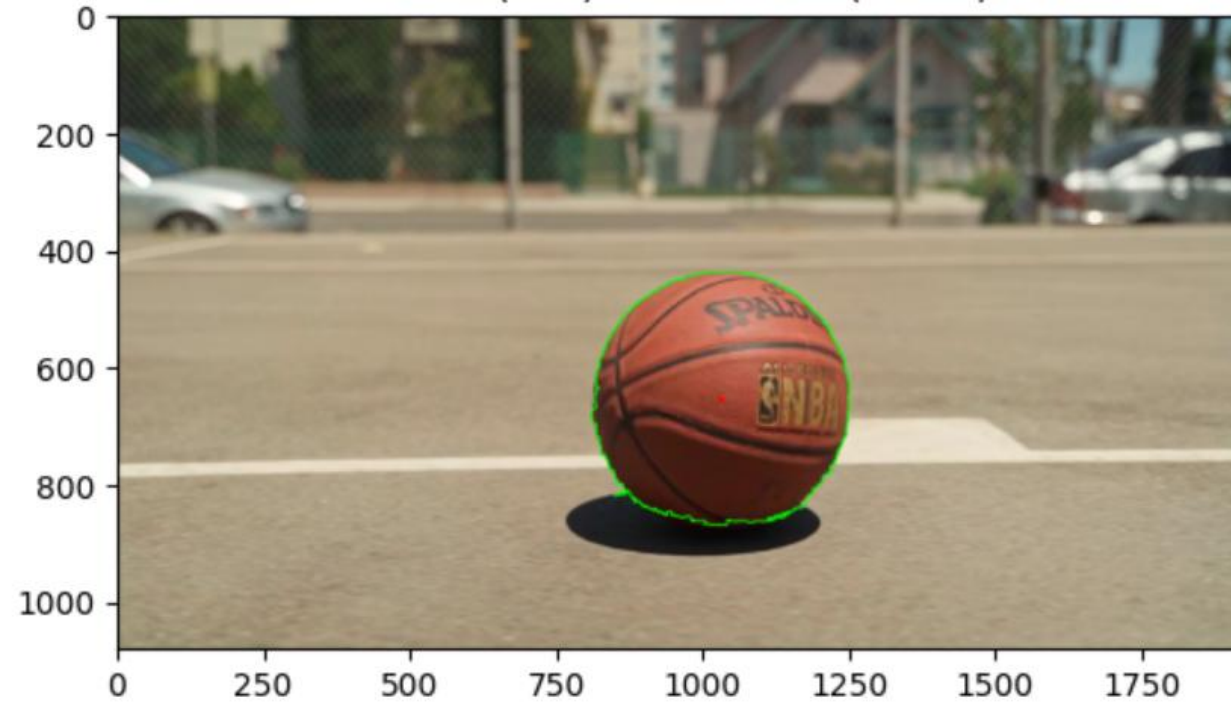
הפונקציה מקבלת את המסיכה

את השיטה (למשל רק קונטורים חיצוניים)

ואת אופן השמירה של הקו (למשל לא כל הפיקסלים שמייצרים אותו אלא רק המינימום הנחוץ)

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

Center(Red) and contour (Green)



מציאת מרכז הקונטור מתבצעת בעזרת הפונקציה

cv2.moments

מה זה moments?

זו פונקציה שמבצעת חישובים גיאומטריים על הקונטור (כמו

מסה, מרכז, שטח)

m_{00} (ה"מסה")

m_{10} ו- m_{01} סכומים משוקללים של מיקומים

באשר מחלקים:

$$cx = m_{10} / m_{00}$$

$$cy = m_{01} / m_{00}$$

מקבלים את מרכז המסה של הצורה – כלומר הנקודה ש"מאזנת"

אותה.

```
M = cv2.moments(contour)
```

```
if M["m00"] != 0:
```

```
    cx = int(M["m10"] / M["m00"])
```

```
    cy = int(M["m01"] / M["m00"])
```

מה זה Kalman Filter?

קלמן פילטר הוא אלגוריתם מתמטי שמבצע **חיזוי + תיקון**.

הוא משמש למעקב אחרי אובייקטים בתנועה – גם כשיש רעש או חוסר ודאות.

איך זה עובד?

בכל רגע:

1. חיזוי – איפה האובייקט אמור להיות (לפי מיקום קודם ומהירות)

2. מדידה – מה רואים בפועל (למשל מהקונטור)

3. תיקון – משקלל בין התחזית למדידה, ומעדכן את המיקום

כך הוא מצליח "לנחש" מיקום גם אם המדידה לא מדויקת – או אפילו חסרה.

למה זה טוב?

• עוקב אחרי אובייקט בתנועה חלקה

• מתקן מדידות עם רעש

• עובד בזמן אמת ובמהירות

נושא	Kalman Filter	למידת מכונה
סוג השיטה	אלגוריתם מתמטי קלאסי	שיטה סטטיסטית שלומדת מנתונים
דרוש אימון מראש?	לא – עובד מיידית בזמן אמת	כן – צריך לאסוף דאטה ולאמן מראש
עיבוד	מהיר מאוד (זמן אמת)	עשוי להיות כבד, תלוי במודל
מתאים ל-	תנועה חלקה, מערכות דינמיות	תבניות מורכבות, זיהוי, תחזית, סיווג
הסתגלות תוך כדי	כן – מתקן את עצמו בכל שלב	לרוב לא (אלא אם מאמנים מחדש)
ידע פנימי על התנועה	כן – משתמש במודל תנועה (מהירות, תאוצה)	לא בהכרח – לומד מהדאטה בלי לדעת את הפיזיקה