

Video Processing



OpenCV היא ספרייה 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, והוא בתובה ב-`C++`, אך היום נפוצה מאוד בשפת `Python` במיוחד בתחום ראייה ממוחשבת, רובוטיקה, בינה מלאכותית ומערכות חכבות.

בעזרת **OpenCV** אנחנו יכולים לקחת תמונה או סרטון ולהפוך אותם לנוטונים:

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")
```

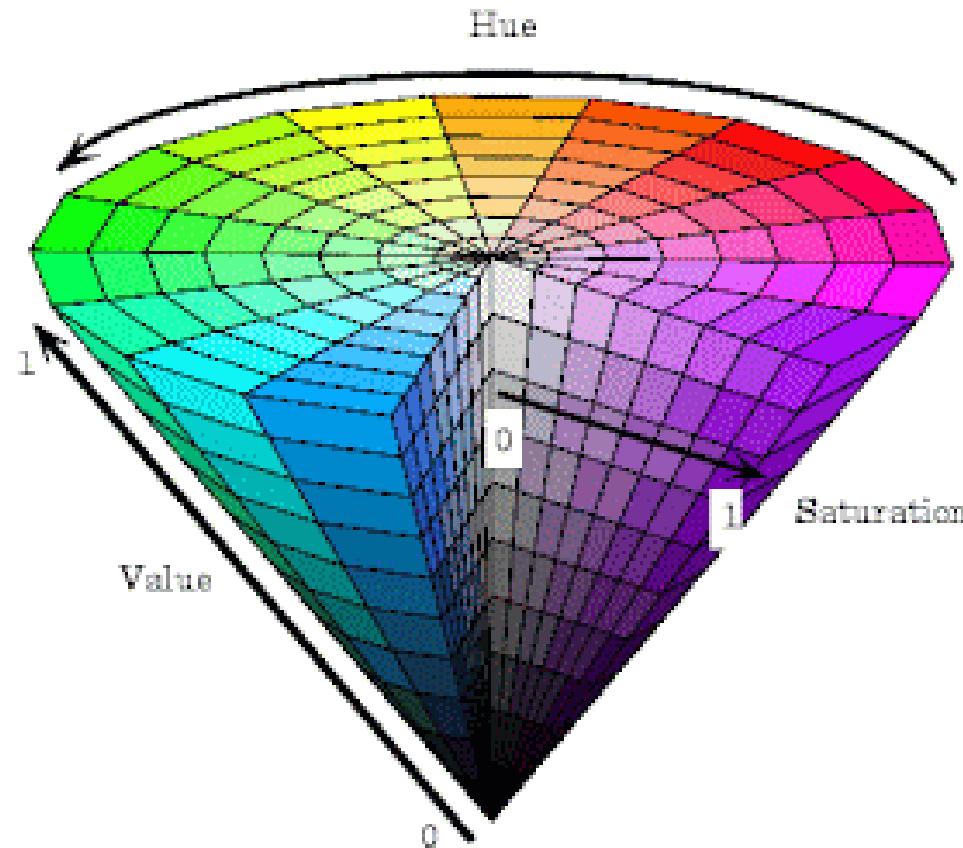
cap הוא כמו "שלט" שמאפשר לנו לגשת לפריים (תמונה בודדות) מתוך הסרט.

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

frame הוא התמונה שנקלטה מהסרטון. ret הוא ערך בוליאני (True/False) שמצוין האם הקריאה הצליחה.



HSV הוא מודל צבע שמנפרק כל צבעו לשלווה רביבים



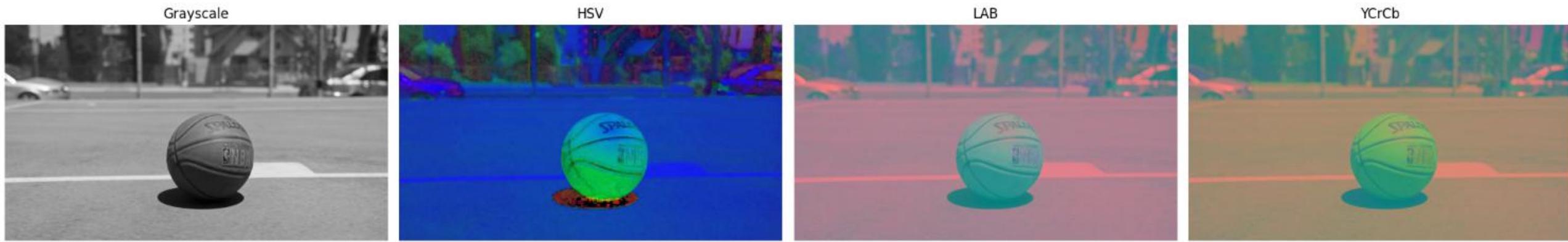
H – Hue – גוון: הצבע עצמו (למשל אדום, ירוק, כחול...) – נמדד בזווית (0–179)

S – Saturation – רוחה: 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 נוספת ומתי משתמשים בהן

מתי נעדיף אותה?	שיטתה
לזיהוי צבעים בתנאי תאורה משתנים	HSV
להצגה, שמירה, ציור, עיבוד בסיסי	RGB
לזיהוי קצוטות, פילטרים, חישובים מהירים	GRAY
כנדריך התאמת מדוקיקת לצבעי ראייה אנושית	LAB

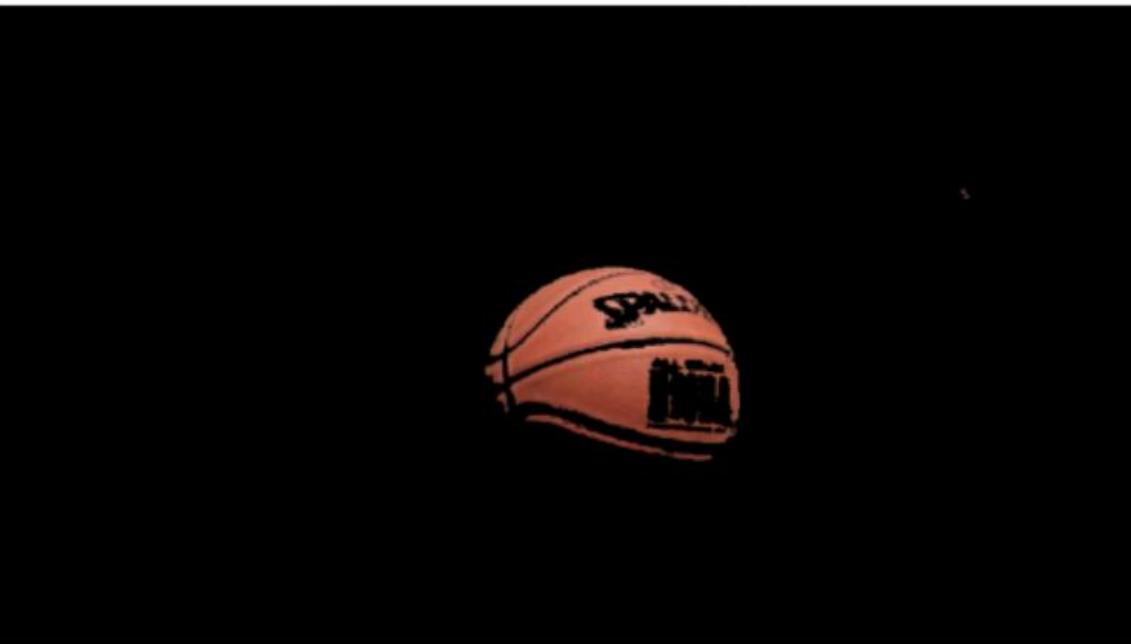


מסיבה היא תמונה בשחור-לבן שמסמנת אילו חלקים בתמונה מעניינים אותנו:

255 (לבן) = אзор פועל / רלוונטי (שחור) = אזור לא רלוונטי / מוסתר

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

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



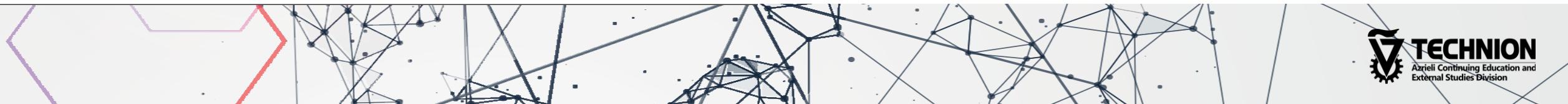
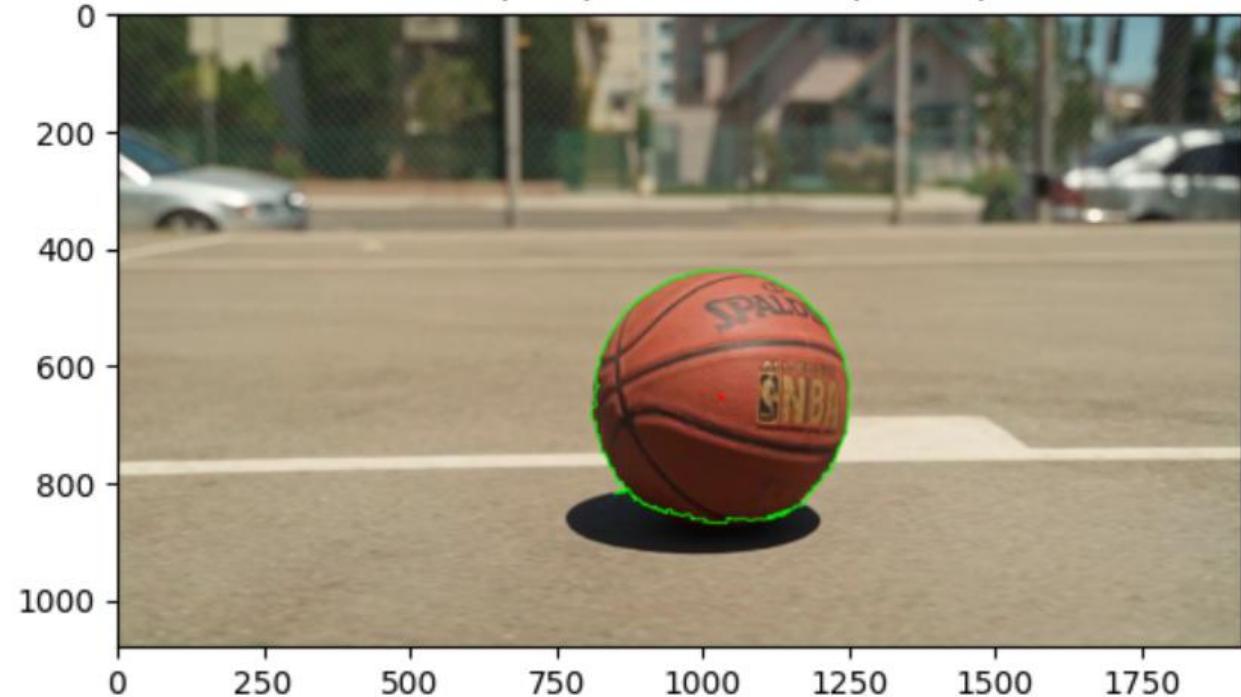
קוונטור הוא קוו שמקיף גבולות של צורה או אובייקט בתמונה.

ב- OpenCV קוונטור הוא בעצם **רשימת נקודות** שמשרטט את קווי המתאר של אזור בולשוי.

למה זה שימושי?

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

Center(Red) and contour (Green)



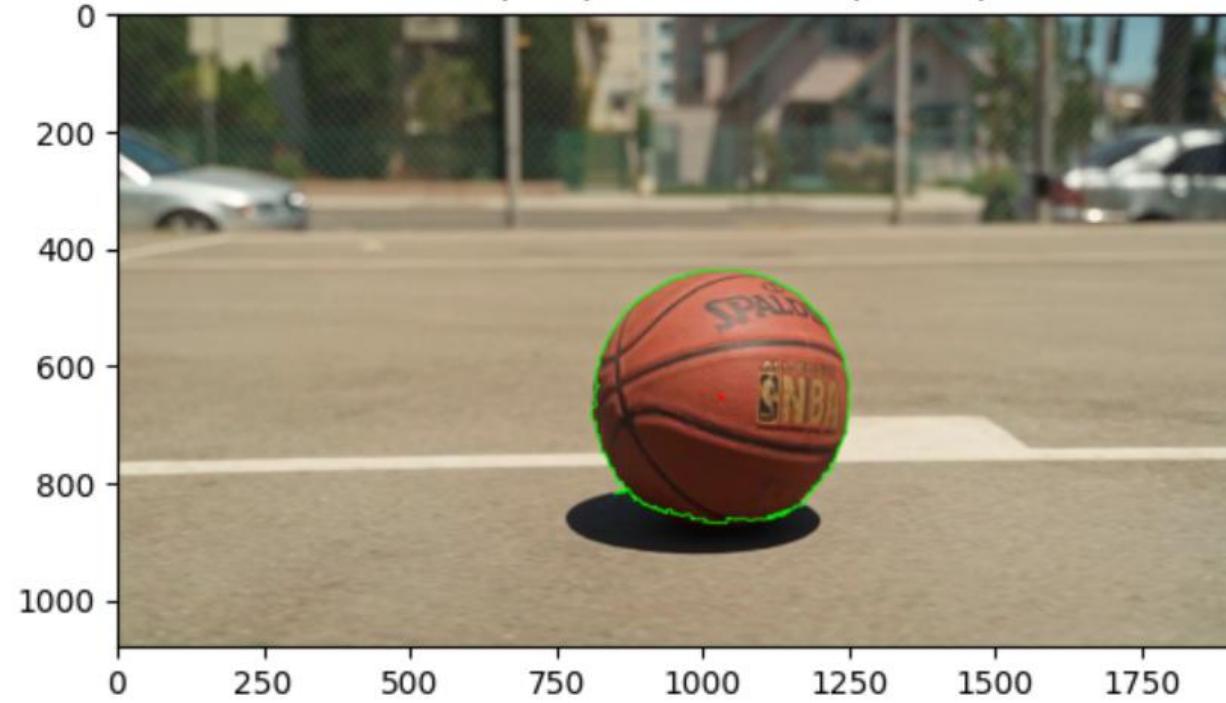
הפונקציה מקבלת את המסיבת

את השיטה (למשל רק קונטורים חיצוניים)

ואת אופן השמירה של הקו (למשל לא בול 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 מתחבצעת בעזרת הפונקציה

`cv2.moments`

מה זה?

זו פונקציה שמבצעת חישובים גיאומטריים על הקונטור (במו
מסה, מרוץ, שטח)

סואזה השטח (ה"מסה")

סואו – וסואם סכומים משוקללים של מיקומים

באשר מחלקים:

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

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

מקבלים את מרכז המסה של הצורה – בלומר 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 ל układ אחראי אובייקטיבים בתנועה – גם בשיש רעש או חוסר מידעות.

איך זה עובד?

בכל רגע:

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

2. מדידה – מה רואים בפועל (למשל מחונטור)

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

בר הוא מצליח "לנחש" מיקום גם אם המדידה לא מדויקת – או אפילו חסורה.

למה זה טוב?

- עוקב אחראי אובייקט בתנועה חלקה

- מתקן מדידות עם רעש

- עובד בזמן אמיתי ובמהירות



נושא	لמידת מכונה	Kalman Filter
סוג השיטה	שיטה סטטיסטיות שלומדת מנתונים	אלגוריתם מתמטי קלאווי
דרוש אימון מראש?	כן – צריך לאסוף>Data ואמן מראש	לא – עובד מיידית בזמן אמיתי
עיבוד	עשוי להיות כבד, תלוי במודל תבניות מורכבות, זיהוי, תחזית, סיווג	מהיר מאוד (זמן אמיתי) תנועה חלקה, מערכות דינמיות
מתאים ל-	להוב לא (אלא אם מאמנים מחדש)	כן – מתקן את עצמו בכל שלב
התגלות תוך כדי ידע פנימי על התנועה	לא בהכרח – לומד מהדата בלבד לדעת את הפיזיקה	כן – משתמש במודל תנועה (מהירות, תאוצה)