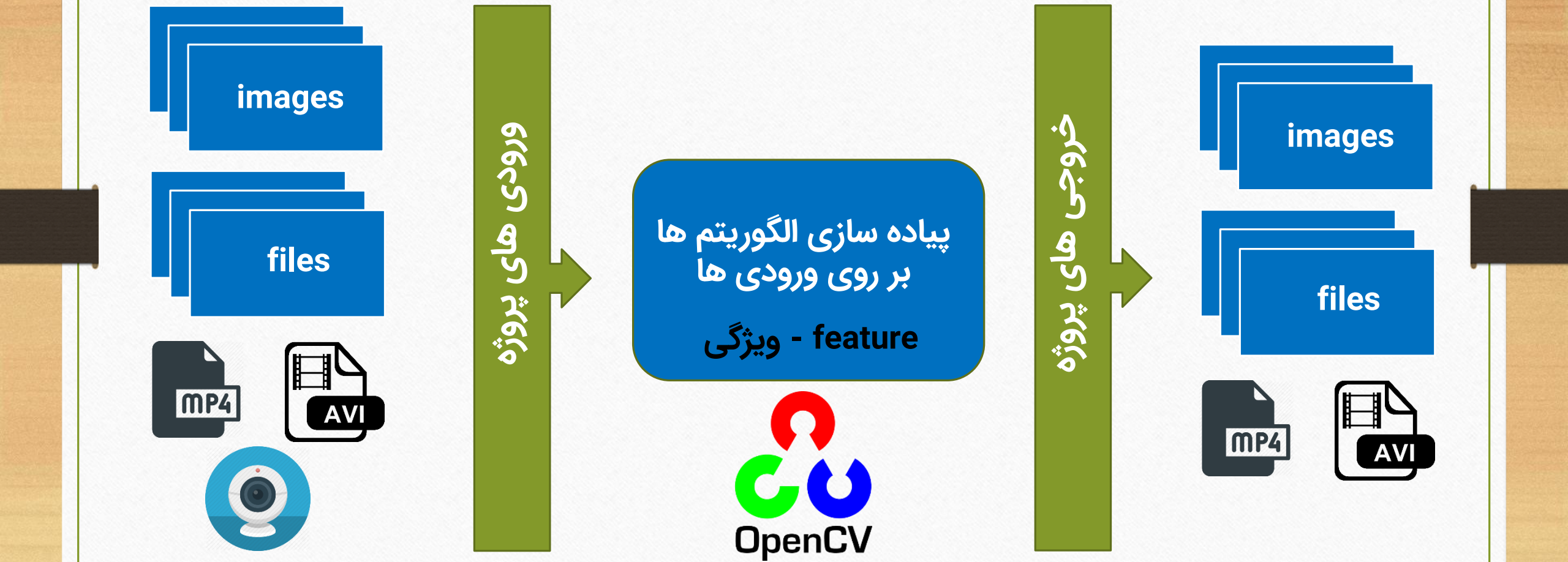
دوره آموزشی بینایی ماشین کاربردی

آکادمی ربوتک – آزمایشگاه تعامل انسان و ربات

جلسه 2 - تصاویر باینری و کار بر روی آنها

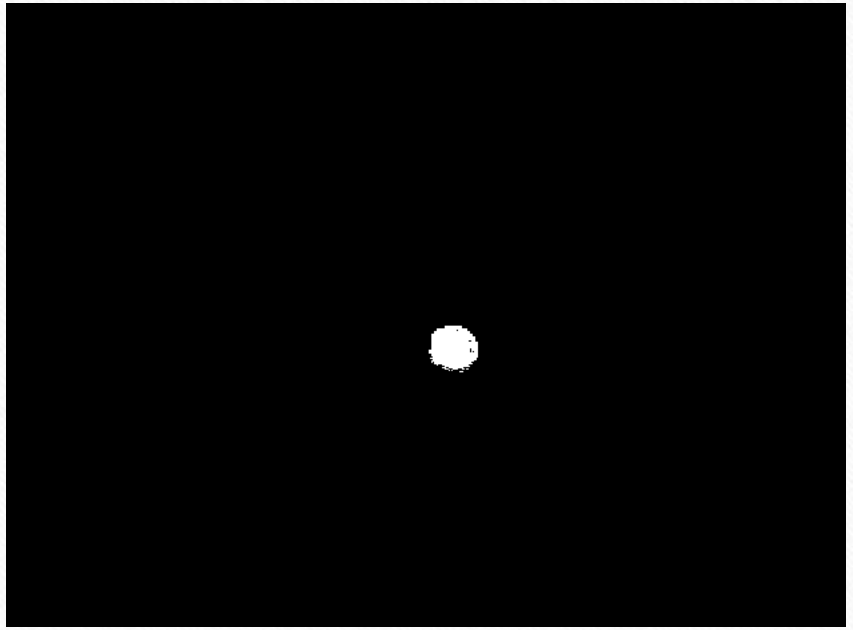
ساختار یک پروژه بینایی ماشین



تشخیص اجسام به کمک رنگ

(ویژگی مورد استفاده : رنگ)

ایده اصلی چیست ؟

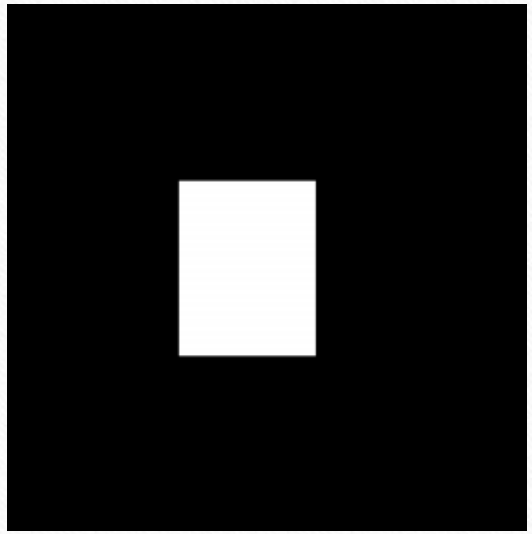


بدست آوردن یک تصویر باینری خوب

آنچه امروز خواهیم گفت :

- تصویر باینری چیست ؟
- روش های ایجاد تصویر باینری
- عملگرهای تصاویر باینری
- محاسبه ویژگی های تصویر باینری

تصاویر باینری چیست؟



$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 255 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

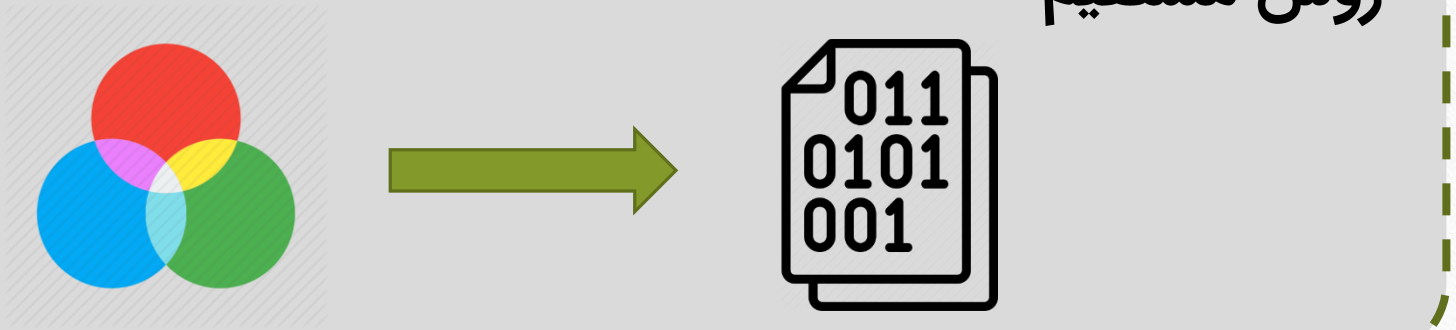
چرا تصاویر باینری مهم هستند ؟

آنچه امروز خواهیم گفت :

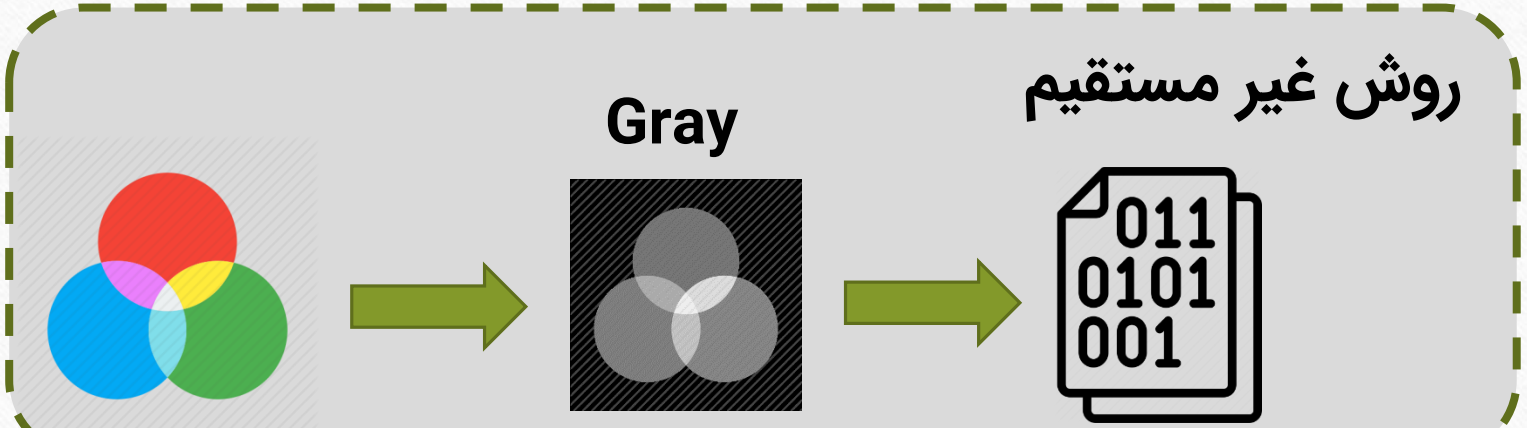
- تصویر باینری چیست ؟
- روش های ایجاد تصویر باینری
- عملگرهای تصاویر باینری
- محاسبه ویژگی های تصویر باینری

روش های ایجاد تصویر باینری

روش مستقیم



روش غیر مستقیم



دستور 21 : ایجاد تصویر Gray (تبدیلات فضای رنگی)

تصویر ورودی

کد تبدیل فضای رنگی

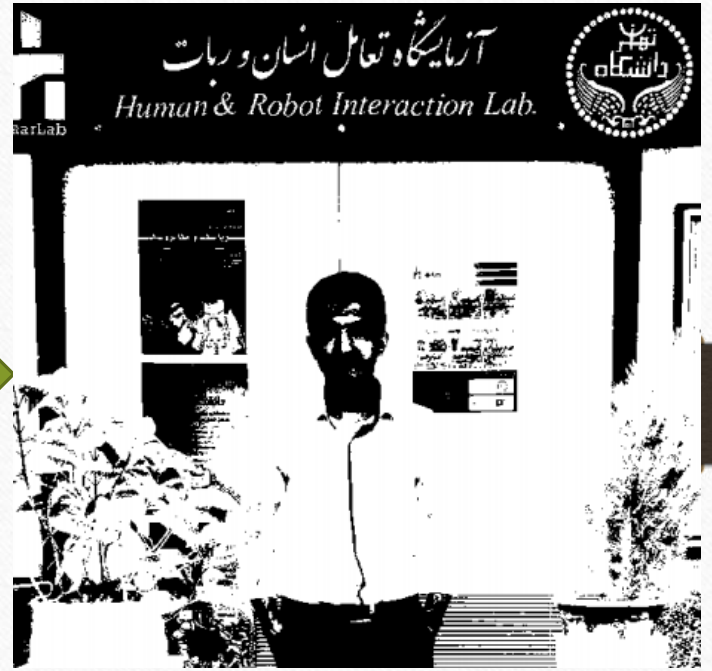
cv2.cvtColor

تصویری با فضای رنگی متفاوت

مثال :

```
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

آستانه گذاری در تصویر



از آستانه گذاری برای تبدیل تصویر Gray به Binary استفاده می شود.

انواع آستانه گذاری ها

آستانه گذاری ساده

از یک مقدار ثابت برای Thresholding استفاده می کند.

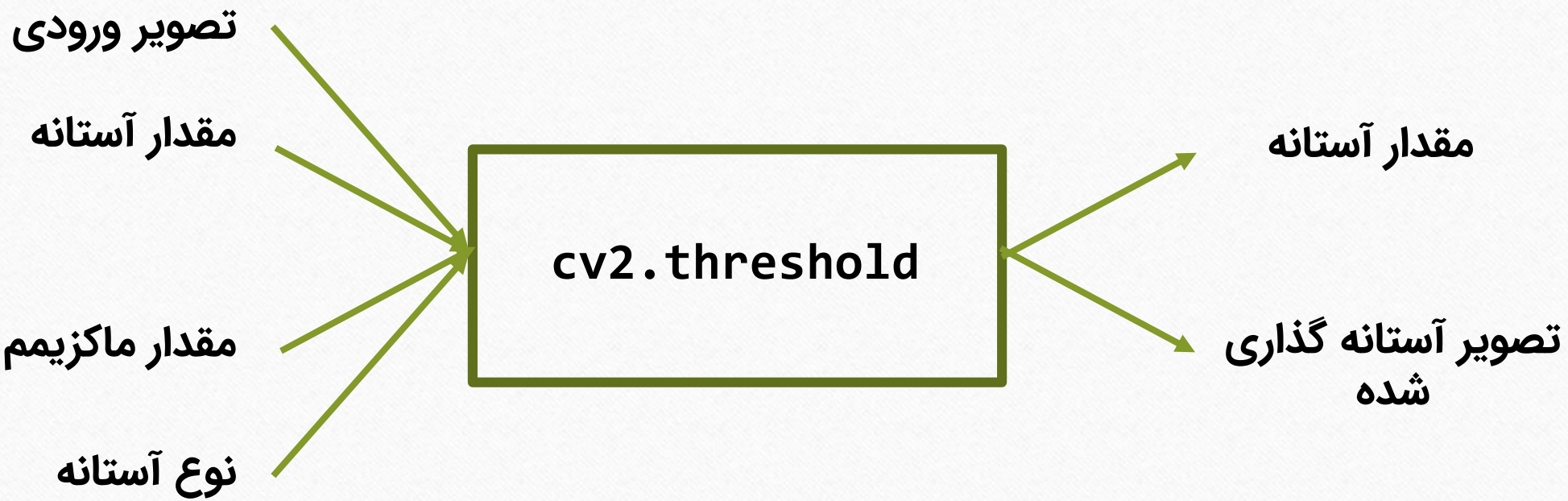
دستور در OpenCV: cv2.threshold

آستانه گذاری adaptive

از یک مقدار متغیر برای Thresholding استفاده می کند.

دستور در OpenCV: cv2.adaptivethreshold

دستور 22 : دستور cv2.threshold



مثال:

```
T, thresh = cv2.threshold(gray, 100, 255, cv2.THRESH_BINARY)
```

انواع متدهای آستانه گیری :

- **cv2.THRESH_BINARY**

$$dst(x,y) = \begin{cases} maxValue & \text{if } src(x,y) > T(x,y) \\ 0 & \text{otherwise} \end{cases}$$

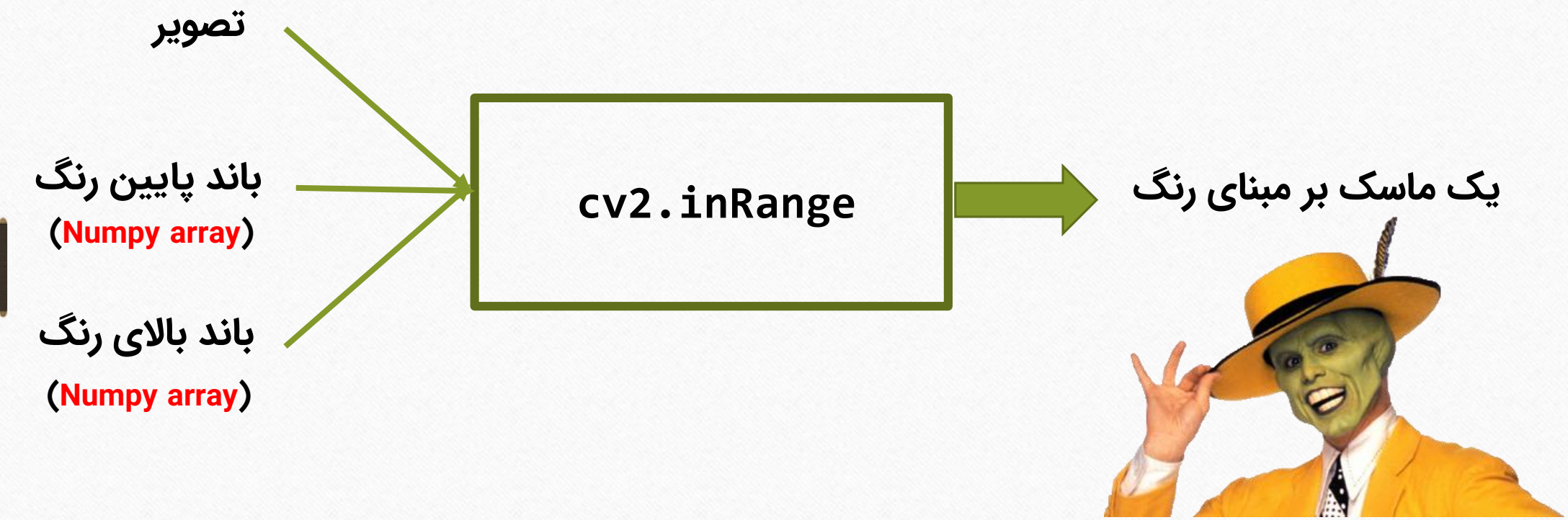
- **cv2.THRESH_BINARY_INV**

$$dst(x,y) = \begin{cases} 0 & \text{if } src(x,y) > T(x,y) \\ maxValue & \text{otherwise} \end{cases}$$

- **cv2.THRESH_TRUNC**

$$dst(x,y) = \begin{cases} threshold & \text{if } src(x,y) > thresh \\ src(x,y) & \text{otherwise} \end{cases}$$

دستور 23 : inRange



مثال :

```
binary_img = cv2.inRange(frame, lower_band, upper_band)
```

مفهوم فضای رنگی

چشم ها را باید شست

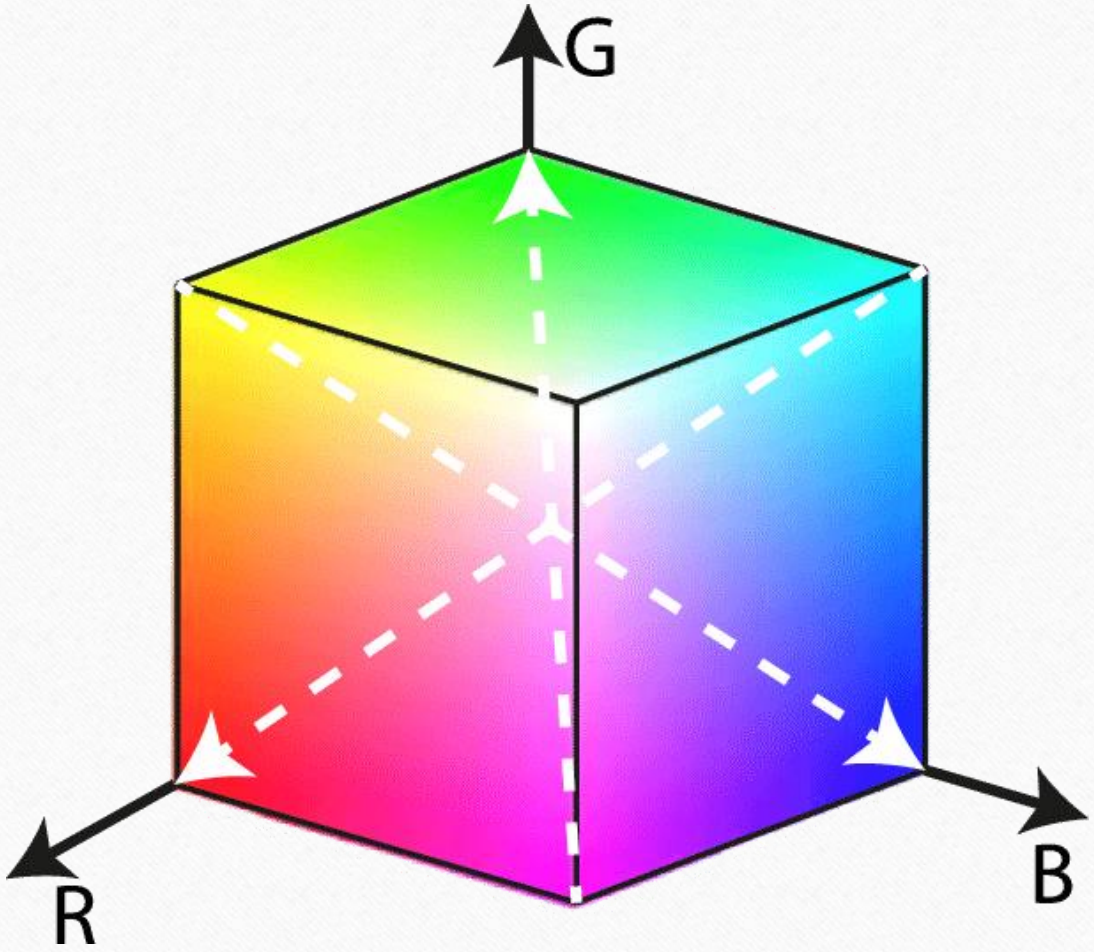
جور دیگر باید دید

- فضای رنگی **مشاهده رنگ ها از نگاهی دیگر** است.

برای **اجرای برخی از دستورات** نیازمند تغییر فضای رنگی هستیم.

فضای رنگی های دیگر **خاصیت های جالبی** دارند.

فضای رنگی RGB

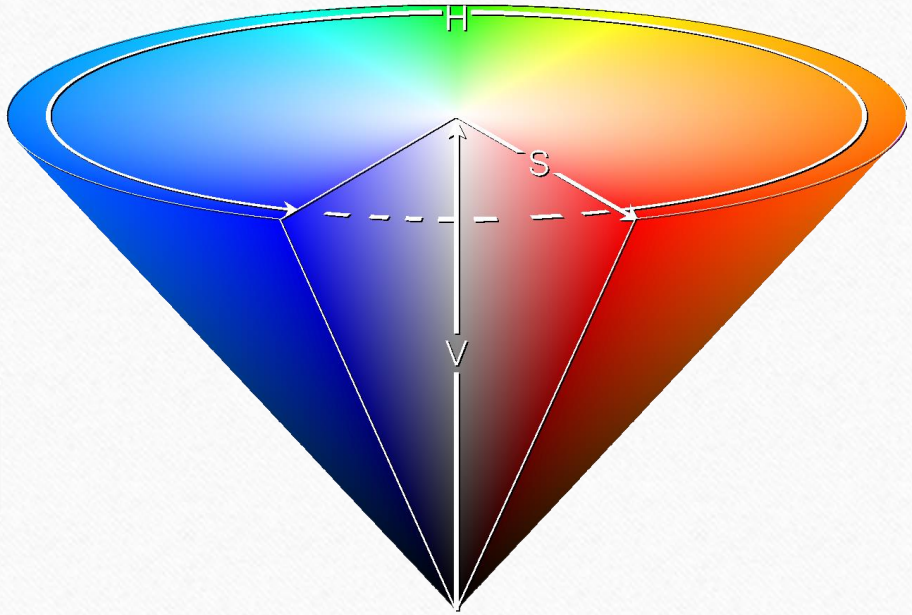


255 > رنگ قرمز > 0

255 > رنگ سبز> 0

255 > رنگ آبی> 0

فضای رنگی HSV



- یک کانال برای رنگ (ترکیب رنگ های RGB و CMYK)
(Hue)
- یک کانال برای میزان رنگی(Saturation)
(Saturation)
- یک کانال برای میزان brightness هر رنگ
(Value)

محدوده Hue :

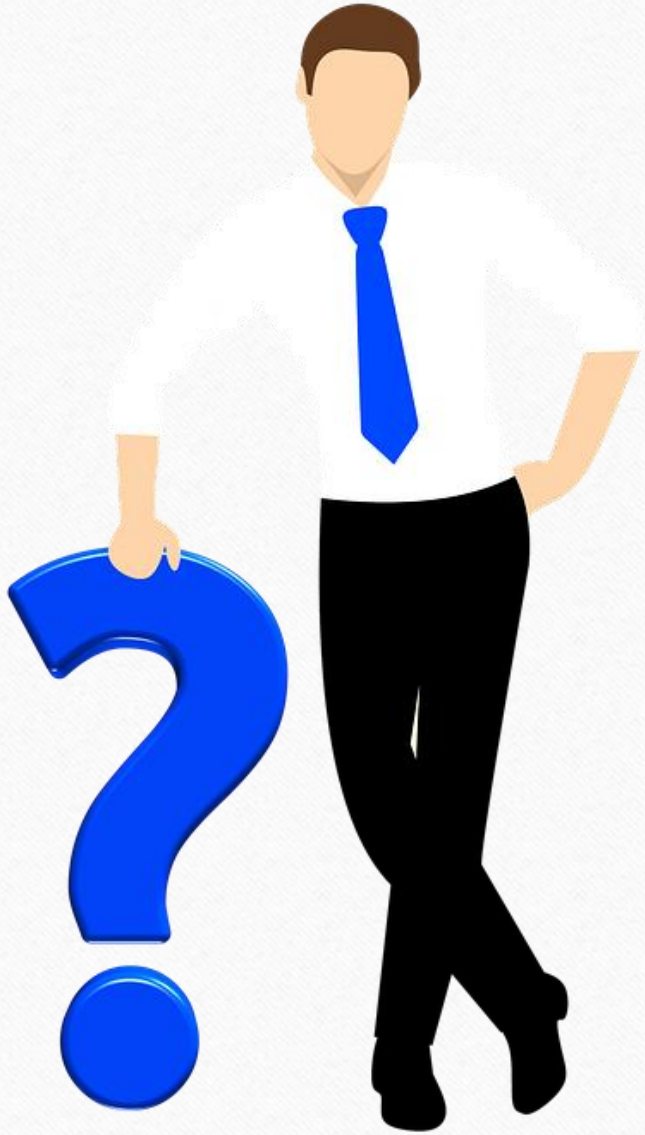
0 - 22	22 - 38	38 - 75	75 - 130	130 - 160	160 - 179

تصویر باینری چیست ؟

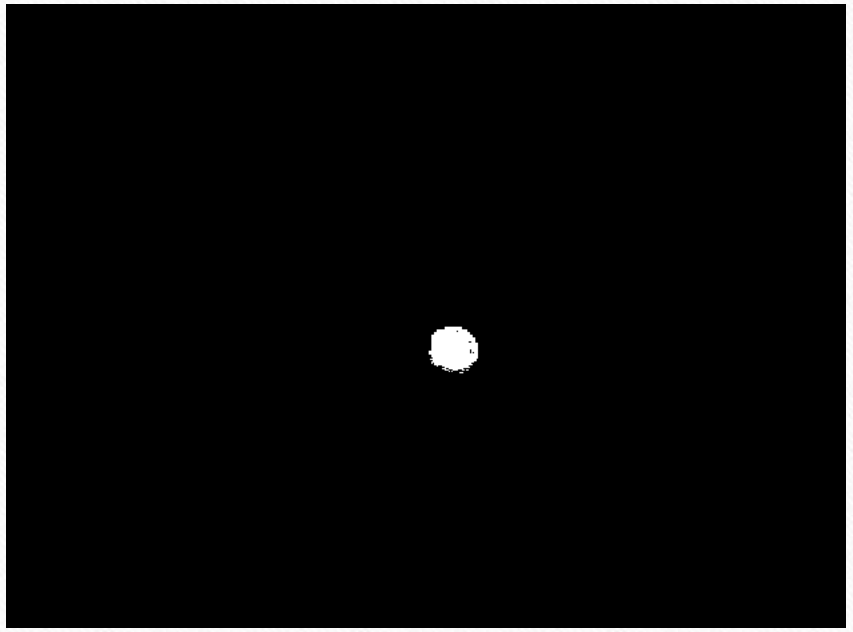
روش های ایجاد تصویر باینری

عملگرهای تصاویر باینری

محاسبه ویژگی های تصویر باینری

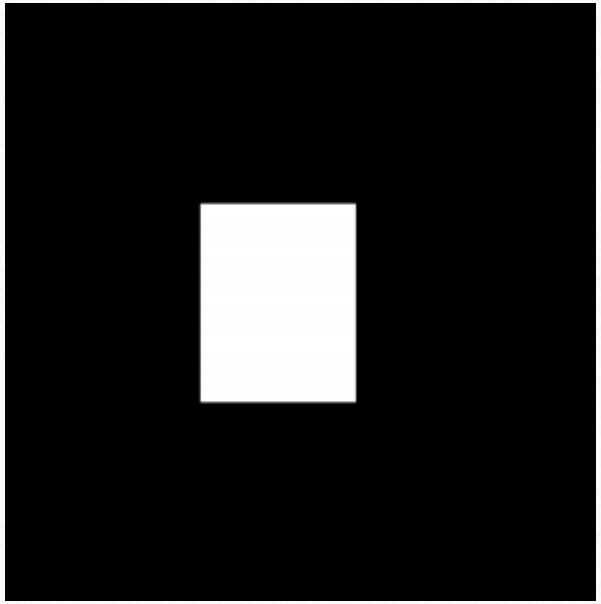


چگونه این دو تصویر را به هم ربط دهیم ؟



عملگرهای بیتی

مفهوم Mask در دنیای تصویر

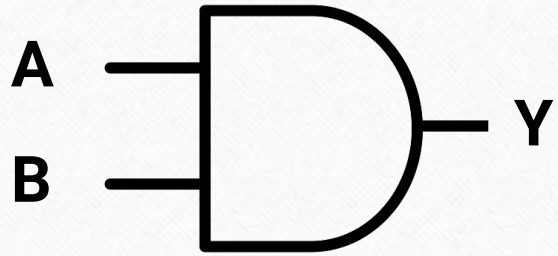


+

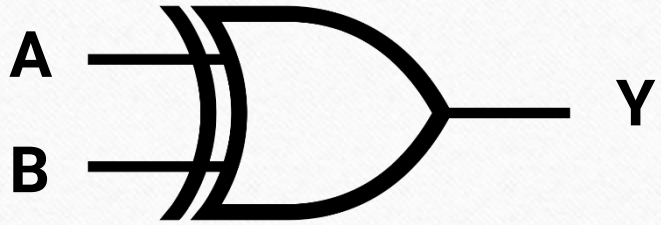
=

نکته 2: ماسک میتواند یک تصویر gray باشد.

نگاهی بر گیت های منطقی



A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

دو تصویر نیز چیزی جز دو آرایه از اعداد نیستند.

عملگر AND عملگر OR

عملگر XOR عملگر NOT

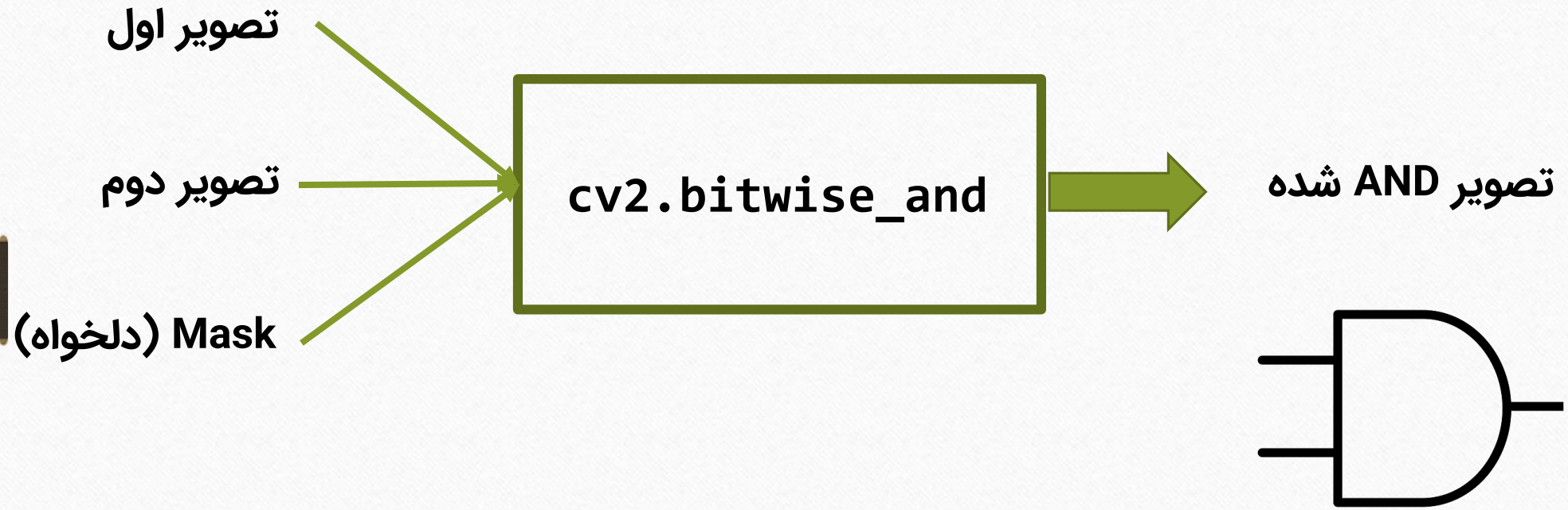
انواع عملگرهای بیتی (bitwise operation)

اندازه دو تصویر باید برابر باشد.

اما یک شرط وجود دارد !

پس میتوانیم از عملگرهایی مشابه گیت های منطقی برای تصاویر استفاده کنیم.

دستور 24 : AND دو تصویر



مثال :

```
and_img = cv2.bitwise_and(img_1, img_2, mask=bin_img)
```

دستور 25 : OR دو تصویر

مثال :

```
or_img = cv2.bitwise_or(img_1, img_2, mask=mask)
```

دستور 26: XOR دو تصویر

تصویر اول

تصویر دوم

Mask (دلخواه)

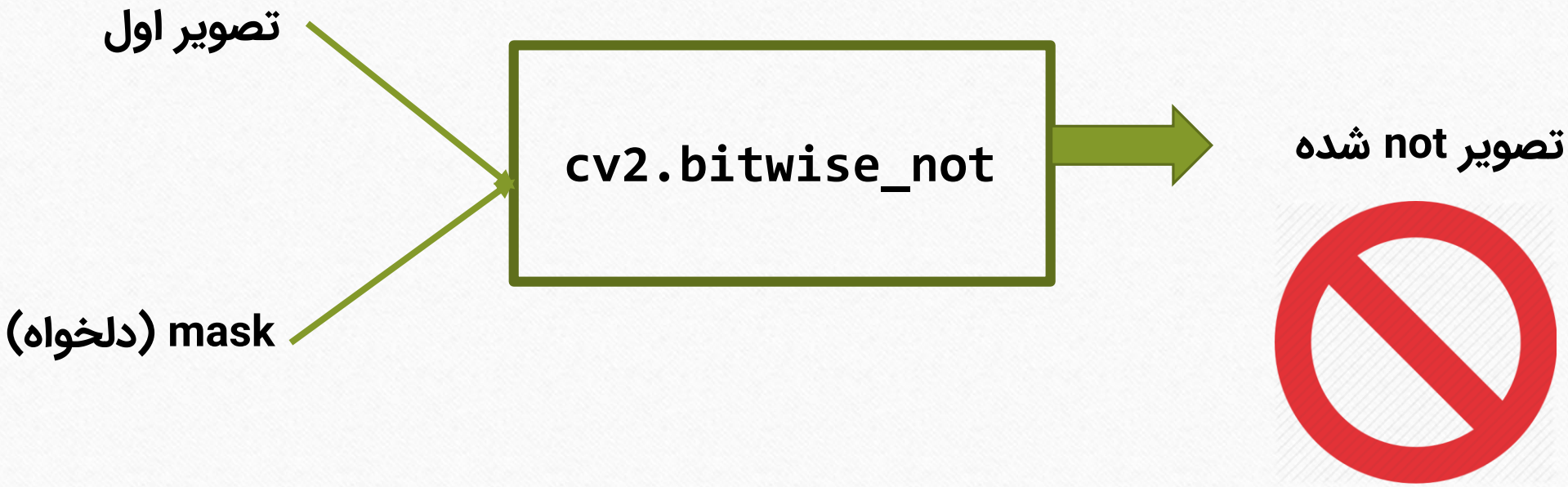
```
cv2.bitwise_xor
```

تصویر XOR شده

مثال :

```
xor_img = cv2.bitwise_xor(img_1, img_2, mask=mask)
```

دستور 27 : NOT یک تصویر



مثال :

```
cv2.bitwise_not(img_1, mask=mask)
```

حل یک مثال : پیدا کردن توپ در تصویر

- خواندن تصویر و تبدیل آن به فضای رنگی HSV
- استفاده از inRange و ساخت Mask مناسب
- استفاده از bitwise_and و پیدا کردن توپ

تمرین : همین کار را برای ویدیویی که در پیوست آورده شده است انجام دهید.

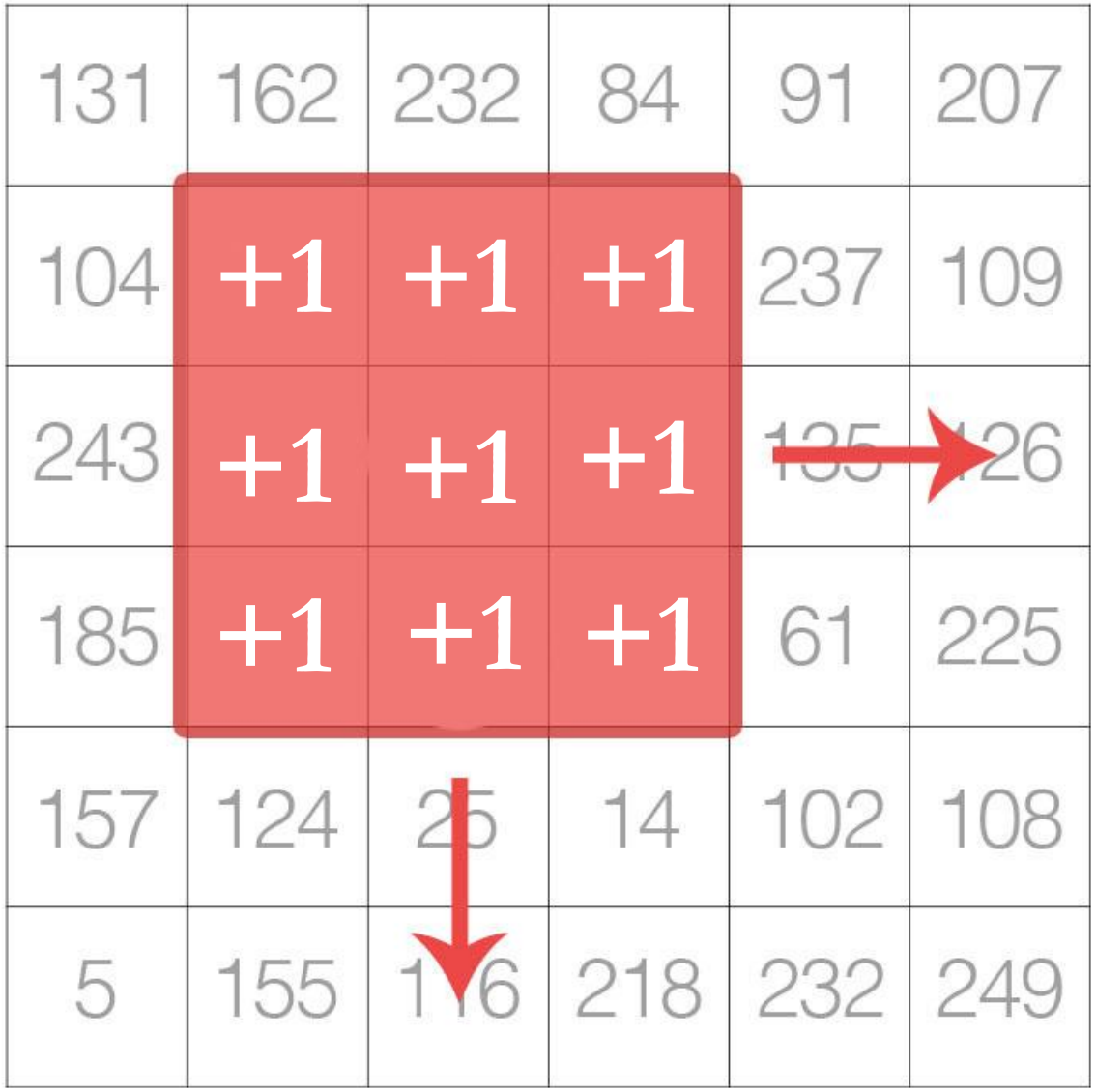
شخصیت هفته : Dr.Adrian Rosebrock

- فارغ التحصیل علوم کامپیوتر دانشگاه مریلند
- نویسنده وبلاگ معروف pyimagesearch
- نویسنده کتابخانه imutils

Learn by Doing and Skip the Theory

تصحیح تصویر باینری

عملگرهای Morphological



عملگری برای بهتر شدن Mask

اعمال بر روی تصاویر باینری

یک کرنل بر روی تصویر
حرکت می کند.

رفع نویز در تصاویر سیاه و سفید

آشنایی با عملگرهای erosion و dilation

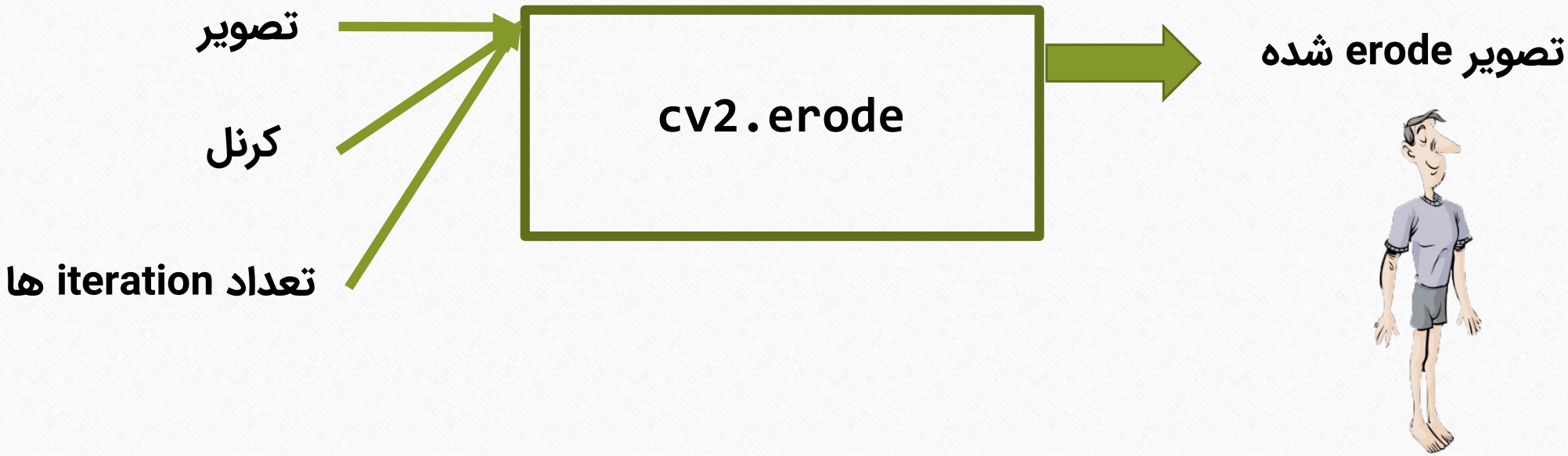
عملگر erosion

مقدار پیکسل تصویر در صورتی یک در نظر گرفته می شود که تمام پیکسل های تصویر در محدوده Kernel ، 1 باشد. در غیر این صورت صفر می شود.

عملگر dilation

مقدار پیکسل تصویر در صورتی یک در نظر گرفته می شود که حداقل یک پیکسل تصویر در محدوده Kernel ، 1 باشد. در غیر این صورت صفر می شود.

دستور 28 : عملگر erosion



مثال :

```
Erosion = cv2.erode(img, kernel, iterations)
```

دستور 29 : عملگر dilation

تصویر

کرنل

تعداد iteration ها

```
cv2.dilate
```

تصویر dilate شده

مثال :

```
Erosion = cv2.dilate(img, kernel, iterations)
```

آشنایی با عملگرهای opening و closing

عملگر opening

ابتدا بر روی تصویر erosion اعمال کنیم
و به دنبال آن dilation اعمال کنیم.

عملگر closing

ابتدا بر روی تصویر dilation اعمال کنیم
و به دنبال آن erosion اعمال کنیم.

دستور 30 : عملگر Opening

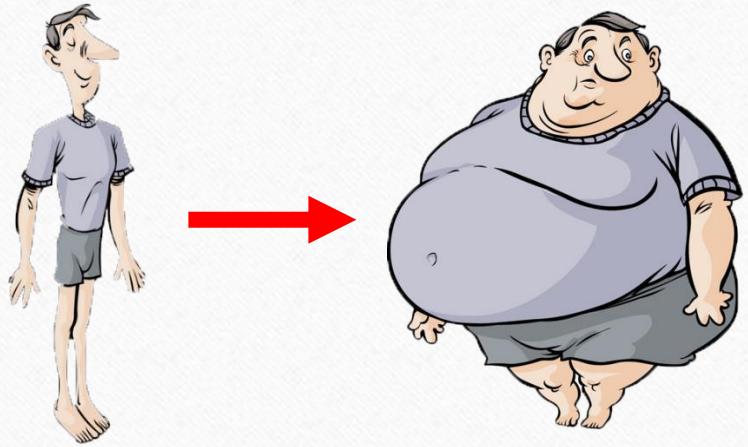
تصویر

پارامتر Open

کرنل

```
cv2.morphologyEx
```

تصویر Open شده



مثال :

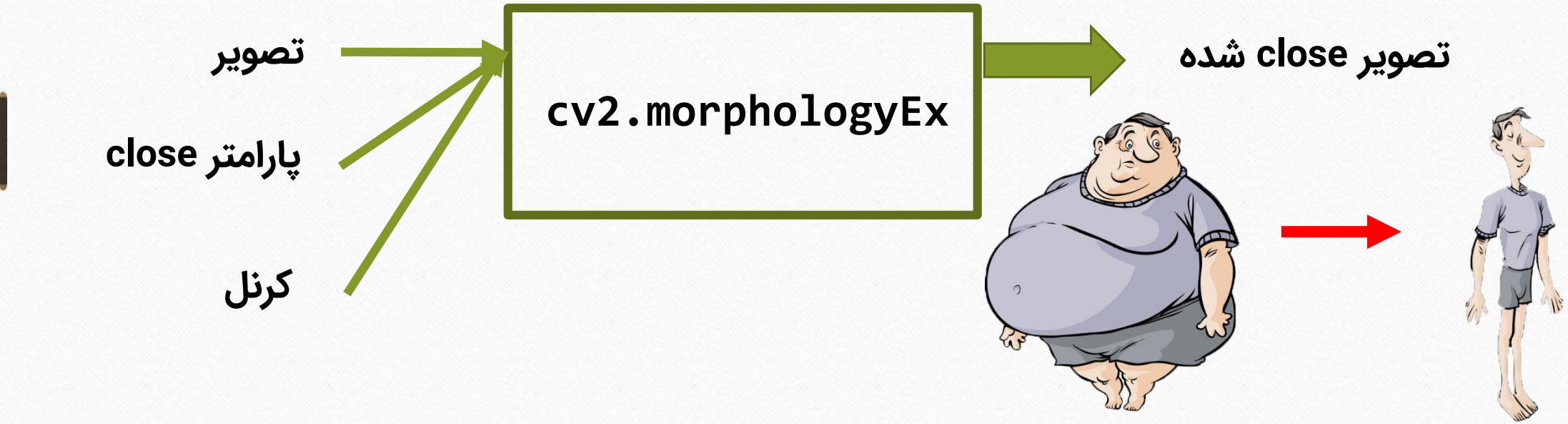
```
cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
```

دستور 31 : عملگر Closing

کاربرد؟

ابتدا بر روی تصویر dilation اعمال کنیم و به دنبال آن erosion اعمال کنیم.

حذف نویز از یک تصویر

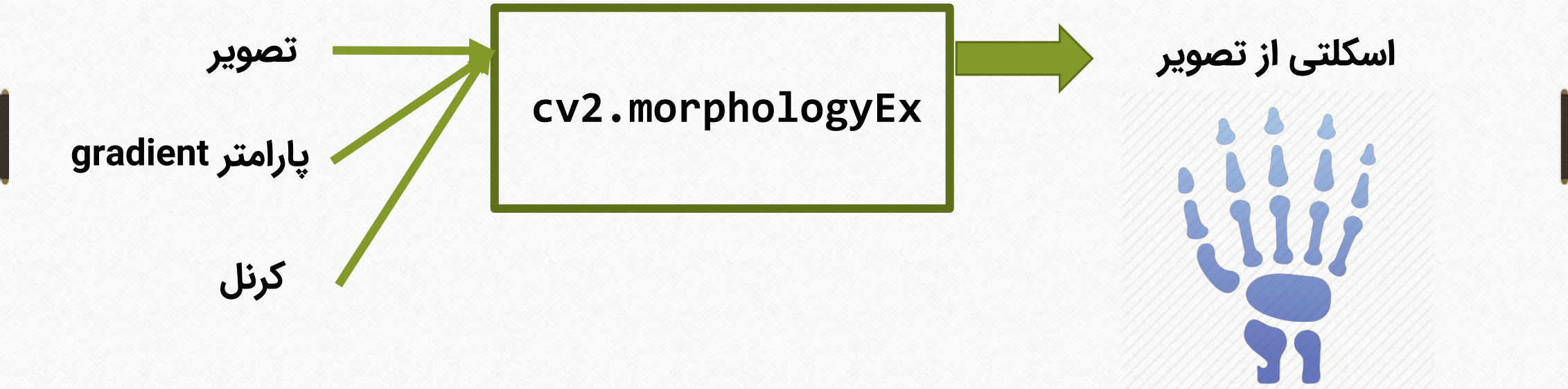


مثال :

```
cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
```

دستور 32 : عملگر Gradient

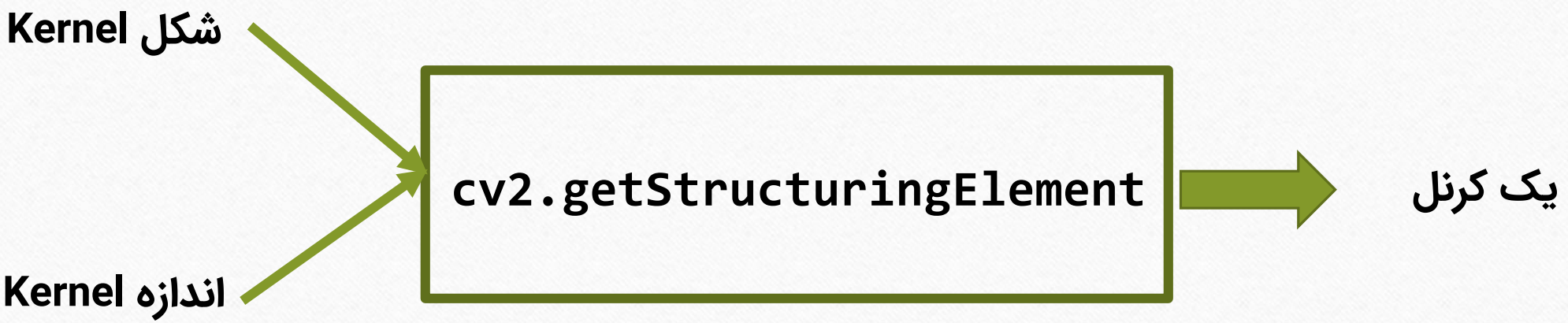
تفریق dilation تصویر و erosion تصویر



مثال :

```
cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)
```

دستور 33 : ساخت Kernel



مثال :

```
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5,5))
```

تصویر باینری چیست ؟

روش های ایجاد تصویر باینری

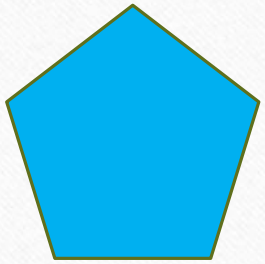
عملگرهای تصاویر باینری

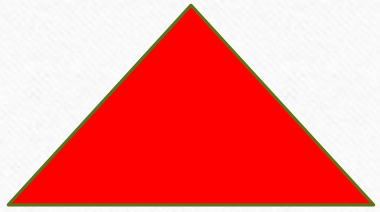
محاسبه ویژگی های تصویر باینری

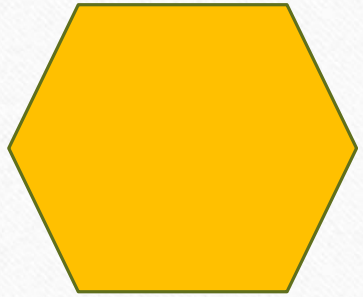
کانتورها

کانتورها مختصات های مرزی نقاطی به هم پیوسته هستند.

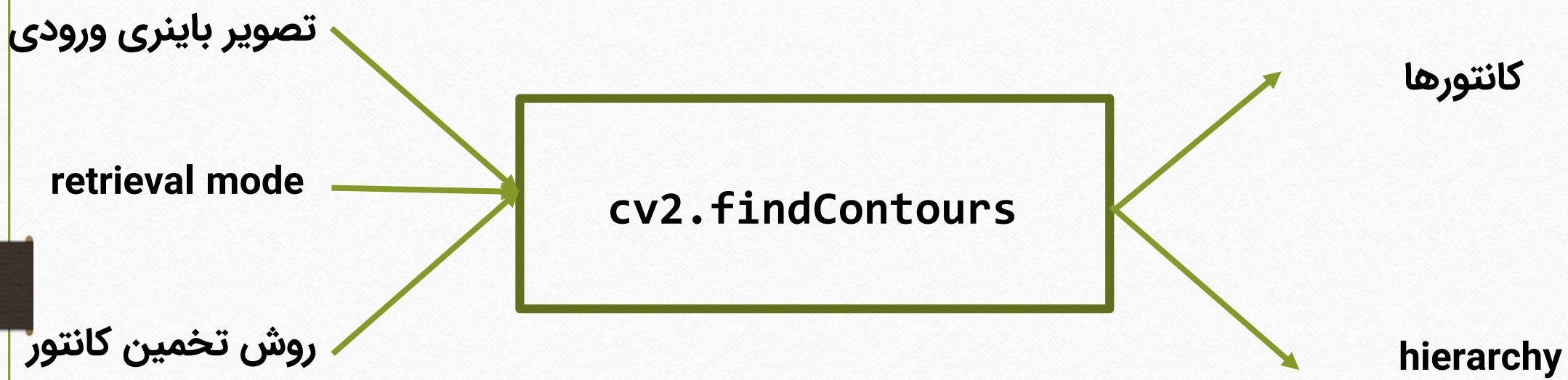
از کانتورها به منظور آنالیز تصاویر باینری استفاده می شود.







دستور 34 : پیدا کردن کانتورها



نکته : کانتورها در واقع یک سری مختصات numpy از مرزهای تصویر باینری است

مثال:

```
cnts, _ = cv2.findContours(inRange_img, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

دستور 35 : رسم کانتورها

تصویر

کانتورها

اندیس کانتور

رنگ

ضخامت

`cv2.drawContours`

رسم کانتور در شکل

مثال:

```
cv2.drawContours(frame, cnts, -1, (0, 255, 0), 3)
```

ویژگی های کانتورها و رسم شکل با کانتور

1. مرکز کانتور
2. محیط کانتور
3. مساحت کانتور
4. محدب بودن یا نبودن
5. تخمین کانتور
6. اشکال اطراف کانتور

دستور 36 :محاسبه مساحت کانتور

کانتور → cv2.contourArea → مساحت کانتور

مثال در OpenCV

```
cnt = contours[0]

Area = cv2.contourArea(cnt)
```

✓ راه دیگر : M['m00']

ویژگی 2 : مرکز کانتور به کمک ممان ها

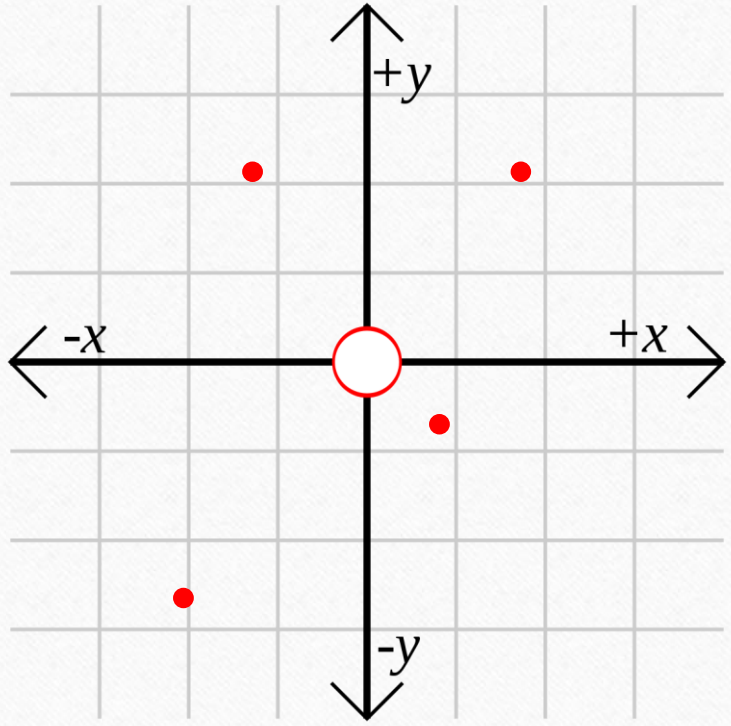
ممان ها یک میانگین وزن دار از Intensity پیکسل ها هستند.

به کمک ممان میتوانیم برخی ویژگی های آماری مثل مساحت کانتور را بیابیم.

میانگین مجموعه ای از داده ها :

$X = [4, \quad 7, \quad 11, \quad 2]$ ⇒ $\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}$

محاسبه میانگین در دو بعد :



$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} \qquad \overline{Y} = \frac{\sum_{i=1}^{n} Y_i}{n}$$

فرمول ممان تصویر باینری

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y)$$

I(x, y) مقدار یک پیکسل ⟵ 0 یا 1

i و j دلخواه از 0 تا 3

i+j مرتبه ممان

مثال ممان مرتبه صفر :

$$M_{00} = \sum_x \sum_y I(x,y)$$

تعداد پیکسل های سفید رنگ، مساحت ناحیه سفید رنگ

محاسبه مرکز کانتور به کمک ممان ها

ممان مرتبه اول

$$M_{10} = \sum_{x}\sum_{y} xI(x,y) = \sum_{i=1}^{n} X_i$$

$$M_{01} = \sum_{x}\sum_{y} yI(x,y) = \sum_{i=1}^{n} Y_i$$

محاسبه مرکز کانتور

$$C_x = \frac{M_{10}}{M_{00}}$$

$$C_y = \frac{M_{01}}{M_{00}}$$

دستور 37 : ممان تصویر

کانتور → `cv2.Moments` → یک دیکشنری که ممان کانتور در آن کد شده است.

محاسبه مرکز یک شی

$$C_x = \frac{M_{10}}{M_{00}}$$

$$C_y = \frac{M_{01}}{M_{00}}$$

نحوه محاسبه در OpenCV

```
cX = int(M['m10']/M['m00'])
cY = int(M['m01']/M['m00'])
```

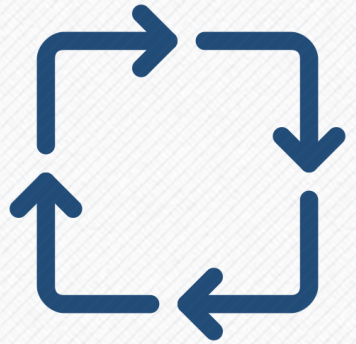
دستور 38: محاسبه محیط کانتور

کانتور

کانتور بسته است یا خیر

cv2.arcLength

محیط کانتور



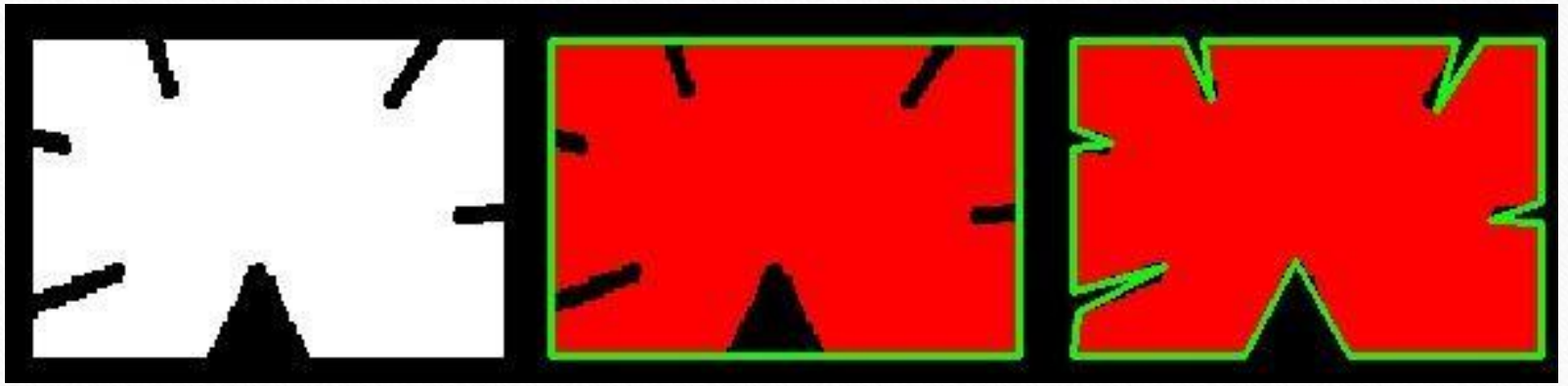
مثال در OpenCV

```
cnt = contours[0]

perimeter = cv2.arcLength(cnt)
```

دستور 39: تخمین کانتور

- کاربرد : در یک تصویر یا ویدیو میتونیم اجسامی که تقریبا شبیه به یک شکل خاص (مثل مستطیل) هستند را به مستطیل تخمین بزنیم و آنها را بیابیم.

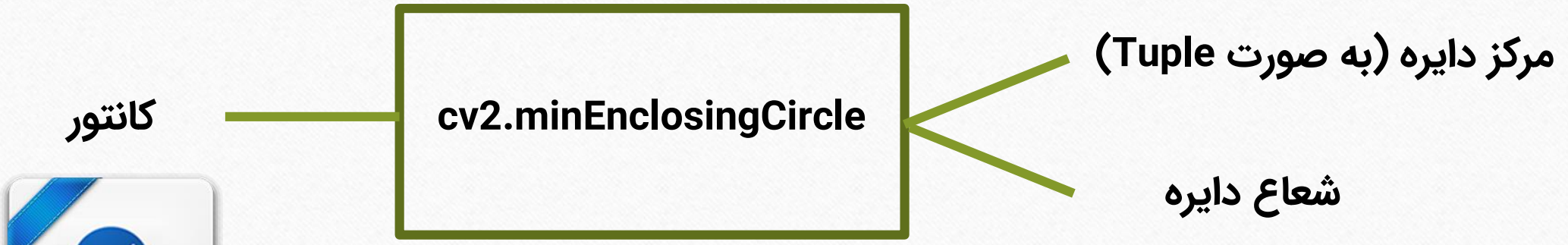


مثال در OpenCV

```
Epsilon = 0.03*cv2.arcLength(cnt, True)

Approx = cv2.approxPolyDP(cnt, epsilon, True)
```

دستور 40: دایره ساده محیط شده بر کانتور



نکته : مرکز و شعاع برای رسم باید به عدد صحیح تبدیل شوند.

مثال:

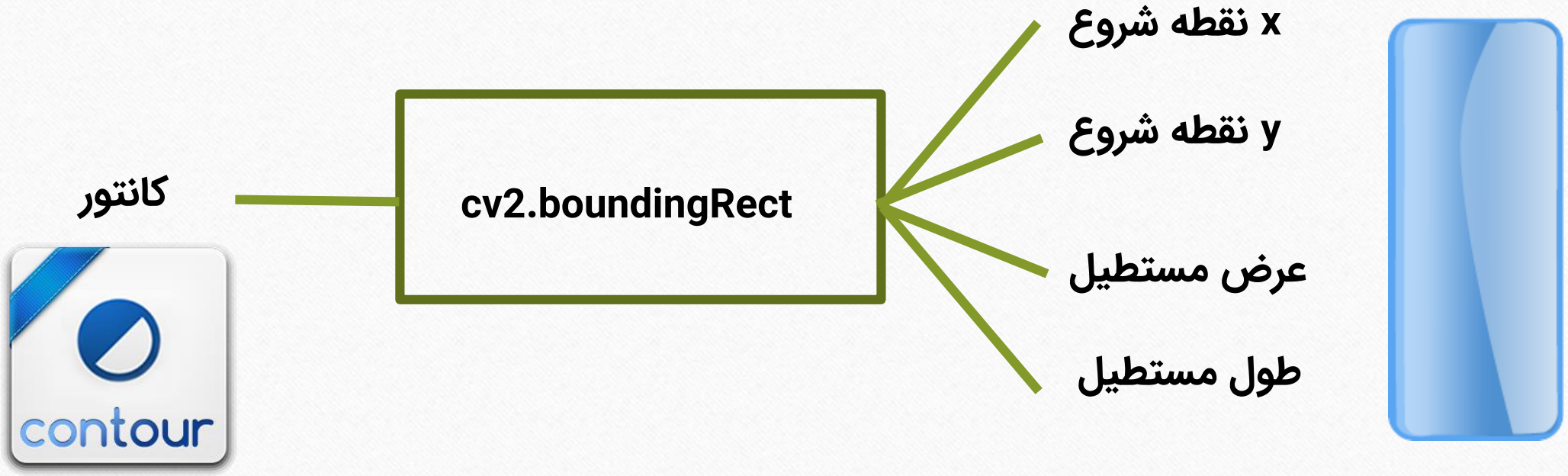
```
(x, y), raduis = cv2.minEnclosingCircle(contour)

center = (int(x), int(y))

raduis = int(raduis)

cv2.circle(frame, center, raduis, (0, 255, 0), 2)
```

دستور 41: مستطیل ساده محیط شده بر کانتور

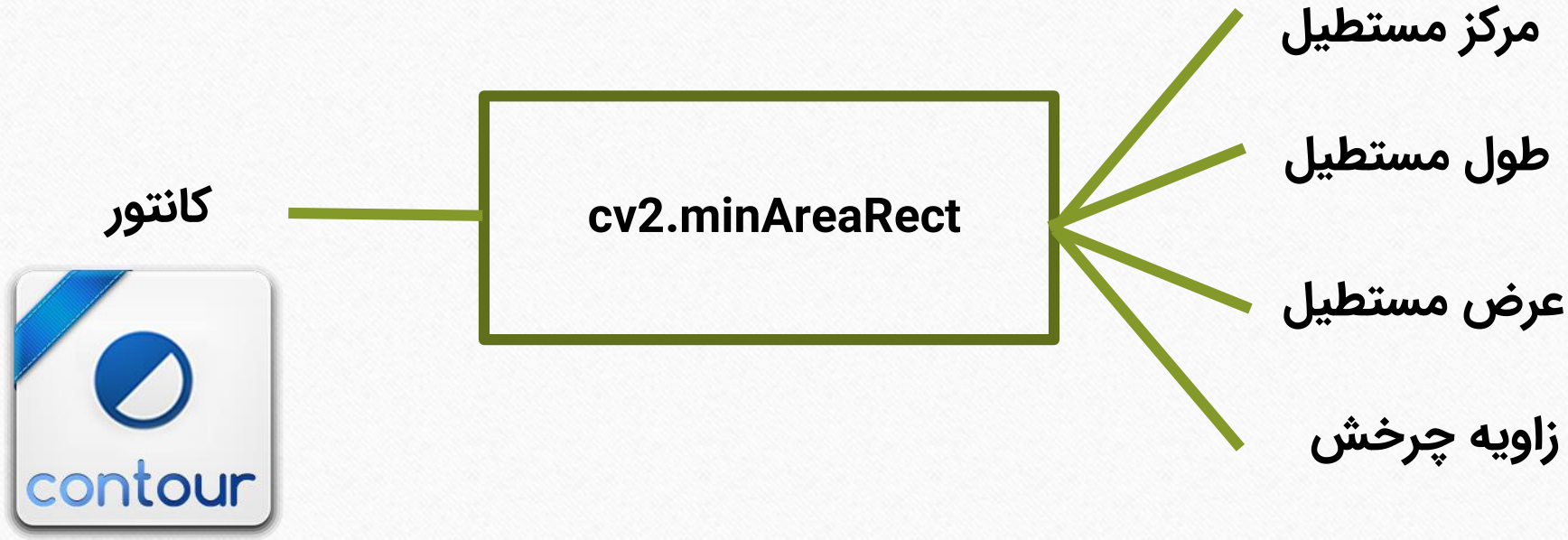


مثال:

```
x, y, w, h = cv2.boundingRect(c)

cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

دستور 42: مستطیل چرخیده محیط شده بر کانتور



مثال در OpenCV

```
rect = cv2.minAreaRect(cnt)
box = cv2.boxPoints(rect)
box = box.astype(int)
cv2.drawContours(img, [box], -1 , (0, 255, 0), 3)
```