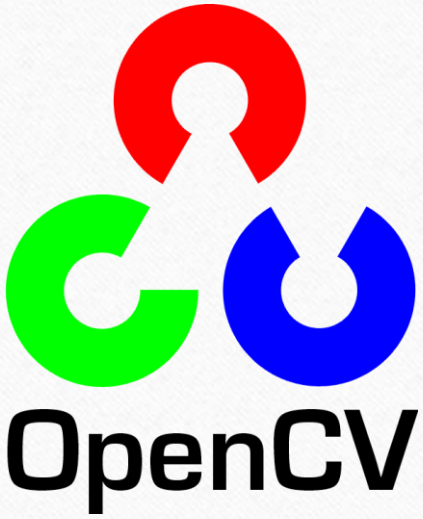
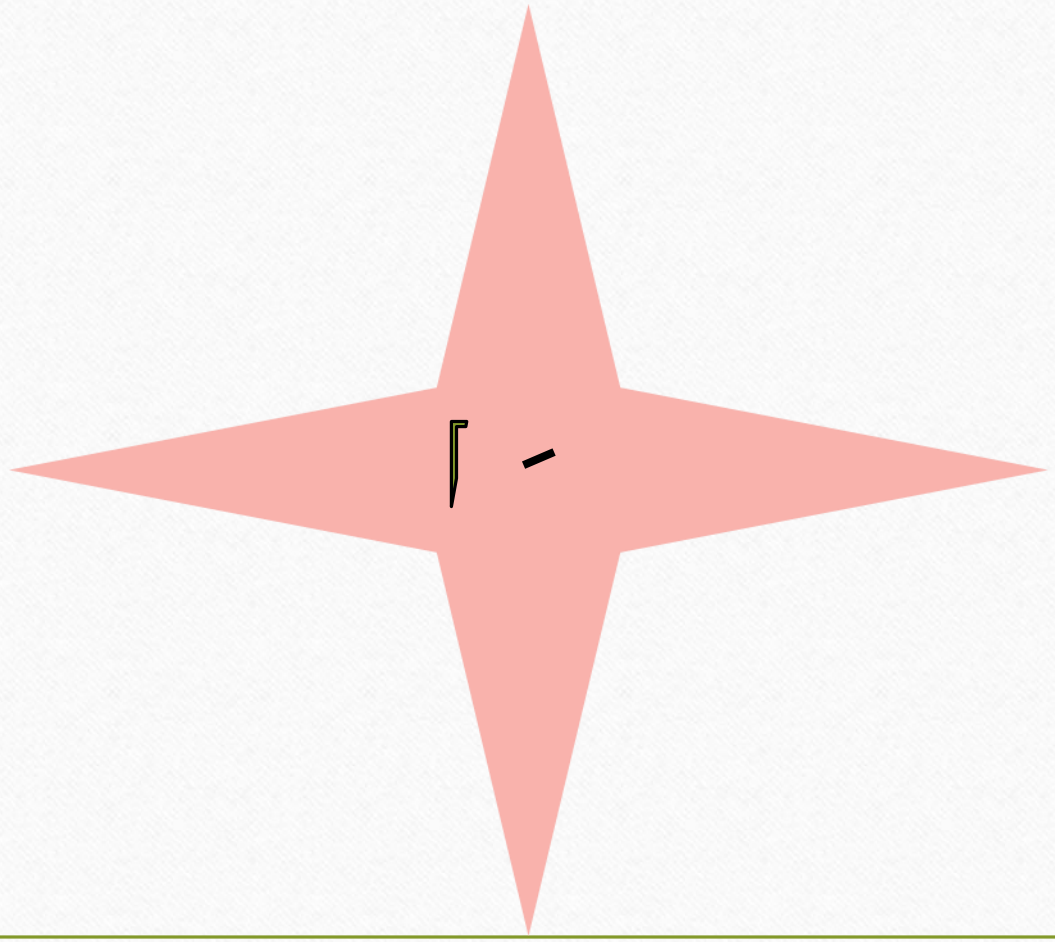
دوره آموزشی بینایی ماشین کاربردی

آکادمی ربوتک – آزمایشگاه تعامل انسان و ربات

جلسه 3 – ادامه کانتورها + لبه یابی



میخواهیم زوایای شکل زیر را پیدا کنیم !



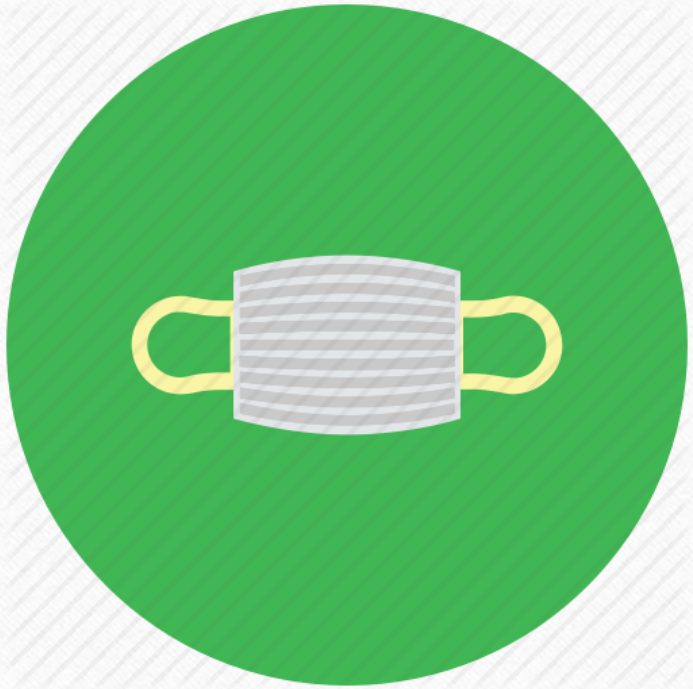
مرحله 1 : ایجاد یک Mask مناسب به کمک HSV

باند پایین = [0 , 33, 63]

باند بالا= [26 , 183, 255]

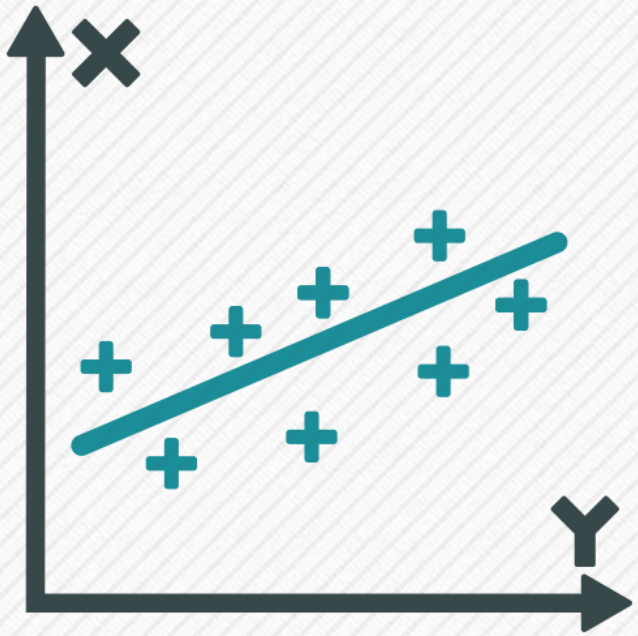
مرحله 2 : تصحیح mask

cv2.erode cv2.dilate



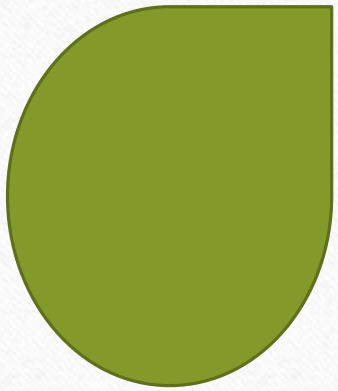
مرحله 3 : محاسبه کانتور و تخمین آن

cv2.approxPolyDP (با دقت بالا)



آشنایی با مفهوم محدب (Convex)

جسم محدب : جسمی است که زاویه ای بیشتر از 180 درجه در خود نداشته باشد.

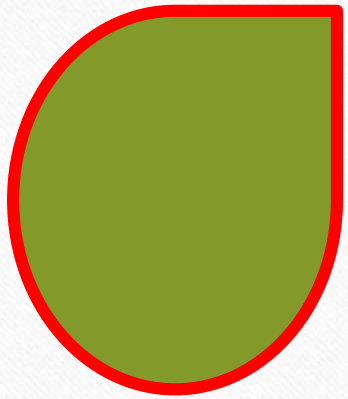


محدب (Convex)

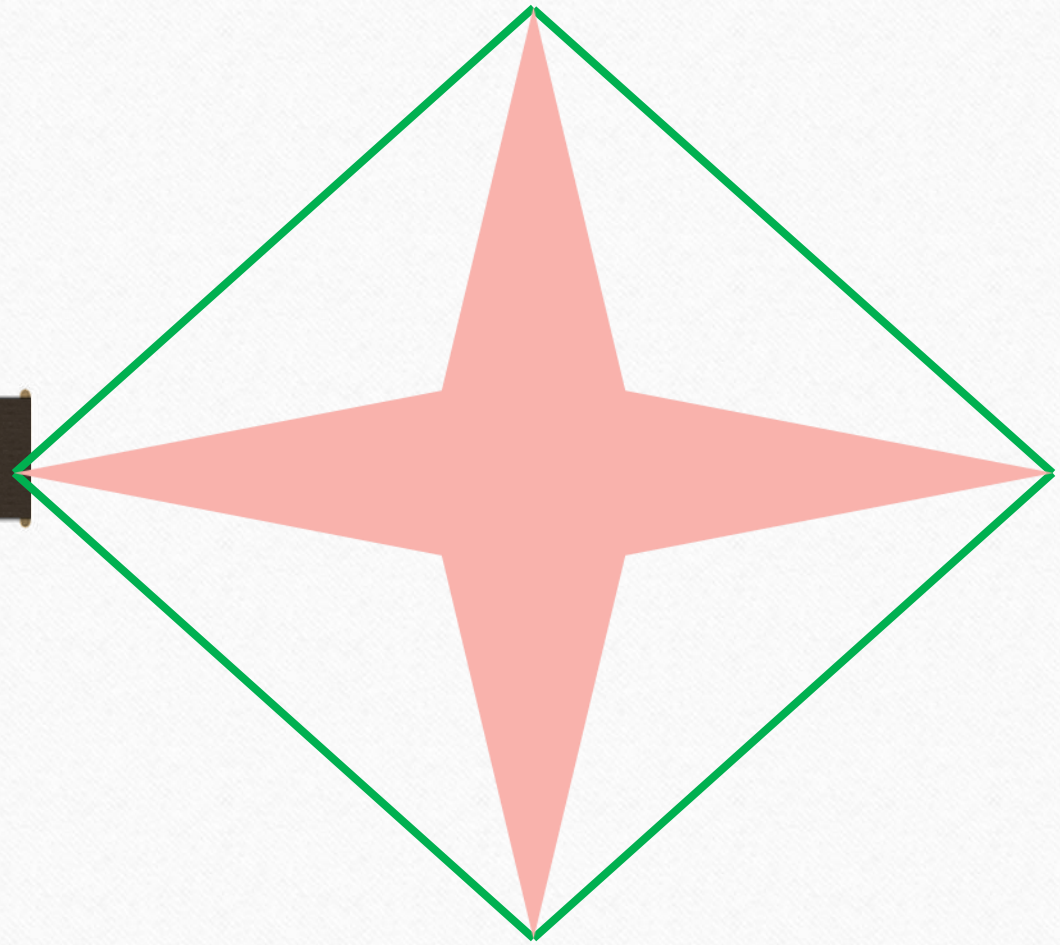
نامحدب (Concave)

Convex Hull

ناحیه محدبی که بر جسم محیط می شود.



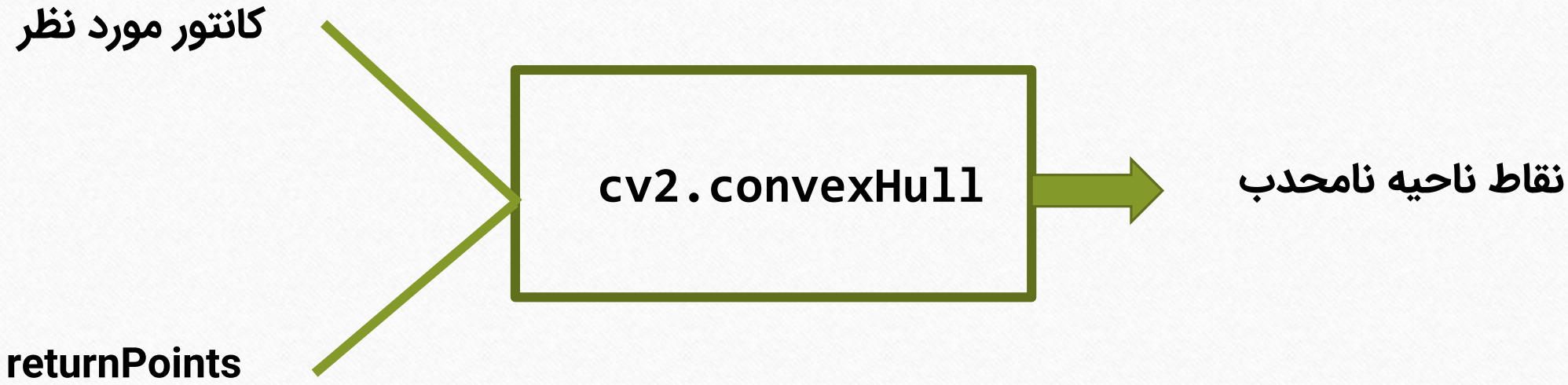
مرحله 4 : محاسبه Convex Hull



میخواهیم یک ناحیه Non Convex در اطراف شکل ایجاد کنیم.

از این ناحیه بعدا برای محاسبه زوایا استفاده می کنیم.

دستور 43 : convexHull



مثال :

```
out_hull = cv2.convexHull(max_cnts)

cv2.drawContours ( img, [out_hull], -1, (0, 255, 0), 2)
```

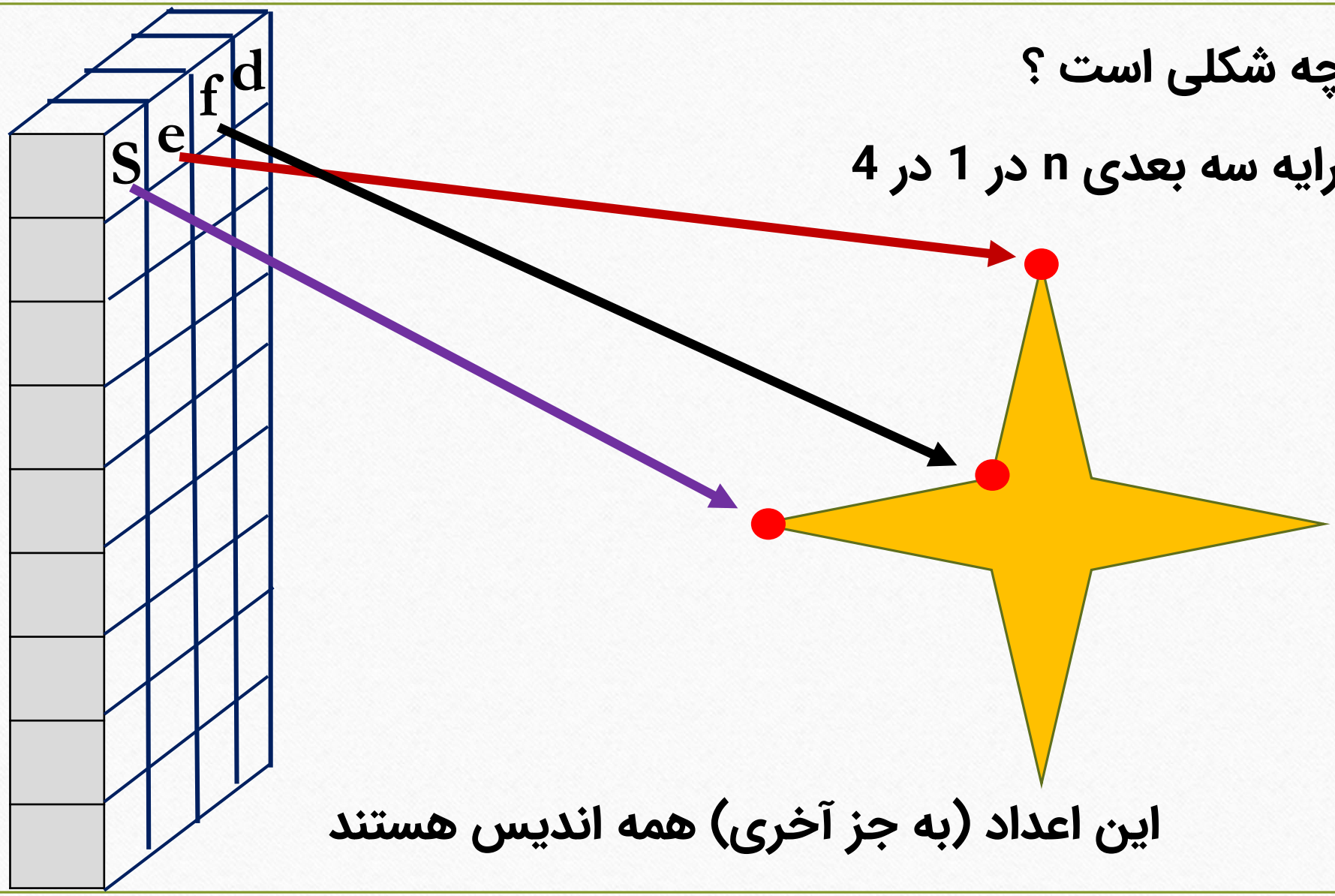
دستور 44: convexityDefects

مثال :

```
defects = cv2.convexityDefects(max_cnts, hull)
```

defects چه شکلی است ؟

یک آرایه سه بعدی n در 1 در 4



این اعداد (به جز آخری) همه اندیس هستند

مرحله 4 : محاسبهConvexity Defects و محاسبه نقاط متناظر کانتور آن

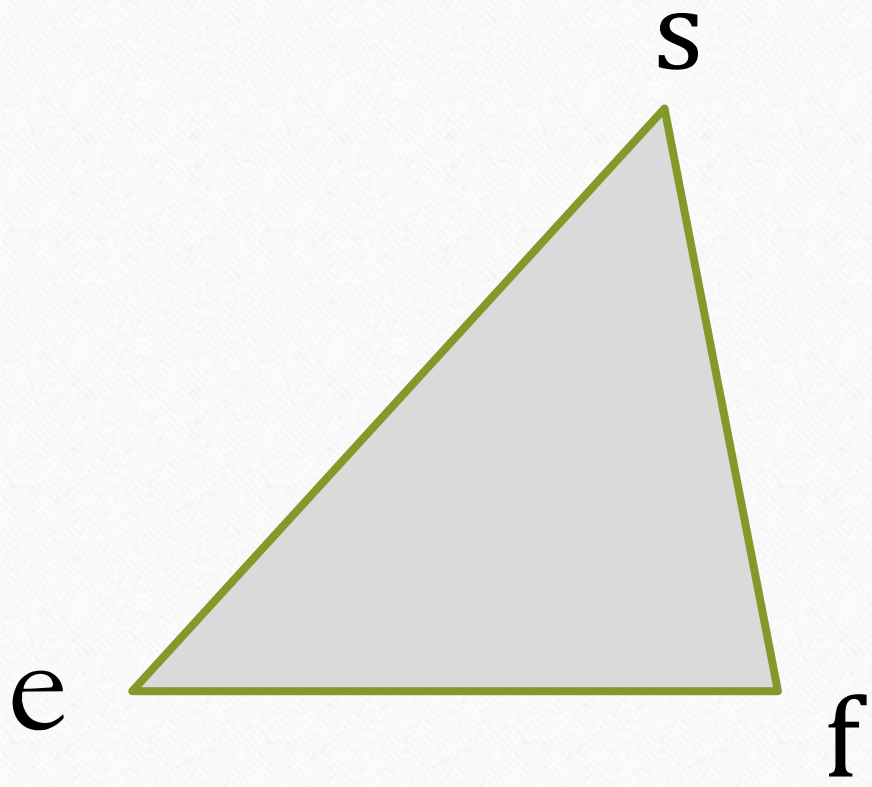
```
for i in range(defects.shape[0]):

    s, e, f, d = defects[i,0]

    start = tuple(max_cnts[s, 0])
    end = tuple(max_cnts[e, 0])
    far = tuple(max_cnts[f, 0])
```

مرحله 6 : محاسبه طول اضلاع مثلث

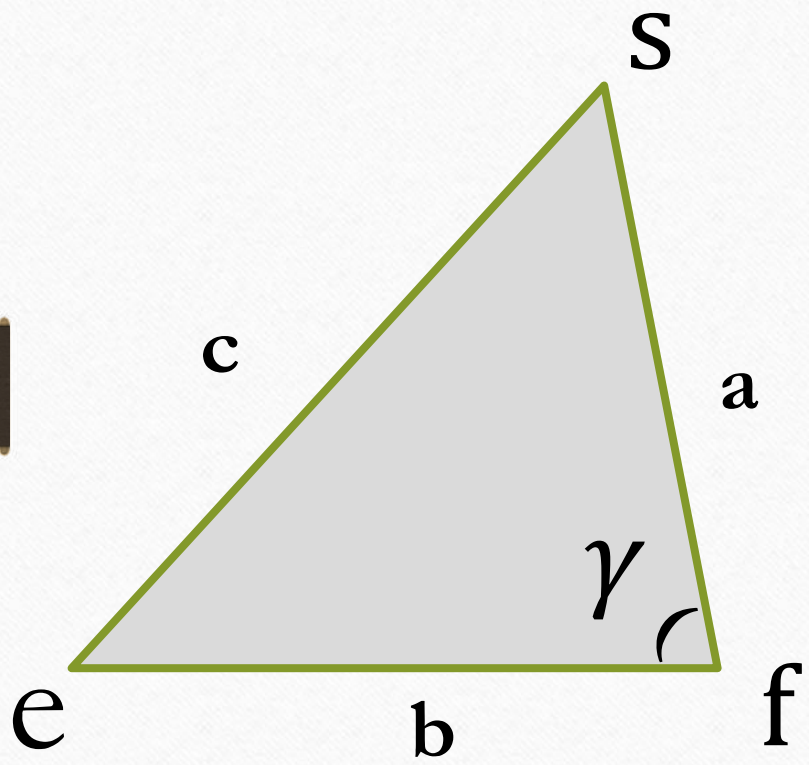
$$d_{sf} = \sqrt{(x_s - x_f)^2 + (y_s - y_f)^2}$$

$$d_{se} = \sqrt{(x_s - x_e)^2 + (y_s - y_e)^2}$$

$$d_{ef} = \sqrt{(x_e - x_f)^2 + (y_e - y_f)^2}$$



مرحله 6 : محاسبه زاویه به کمک قانون کسینوس ها



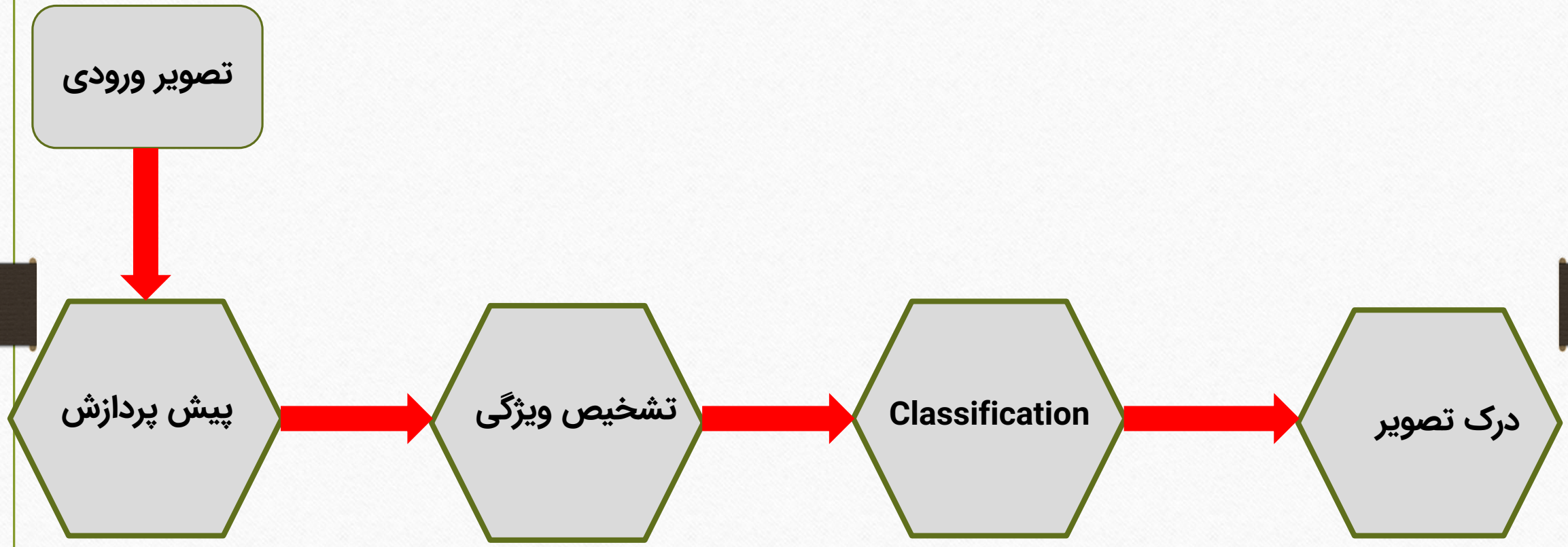
$$\gamma = \frac{a^2 + b^2 - c^2}{2ab}$$

تشخیص ژست دست (Hand Gesture Recognition)

تعداد انگشت = تعداد زوایای کمتر از 90 + 1

بخش دوم

الگوریتم های تشخیص ویژگی


تصویر ورودی
پیش پردازش
تشخیص ویژگی
Classification
درک تصویر

بخش اول:

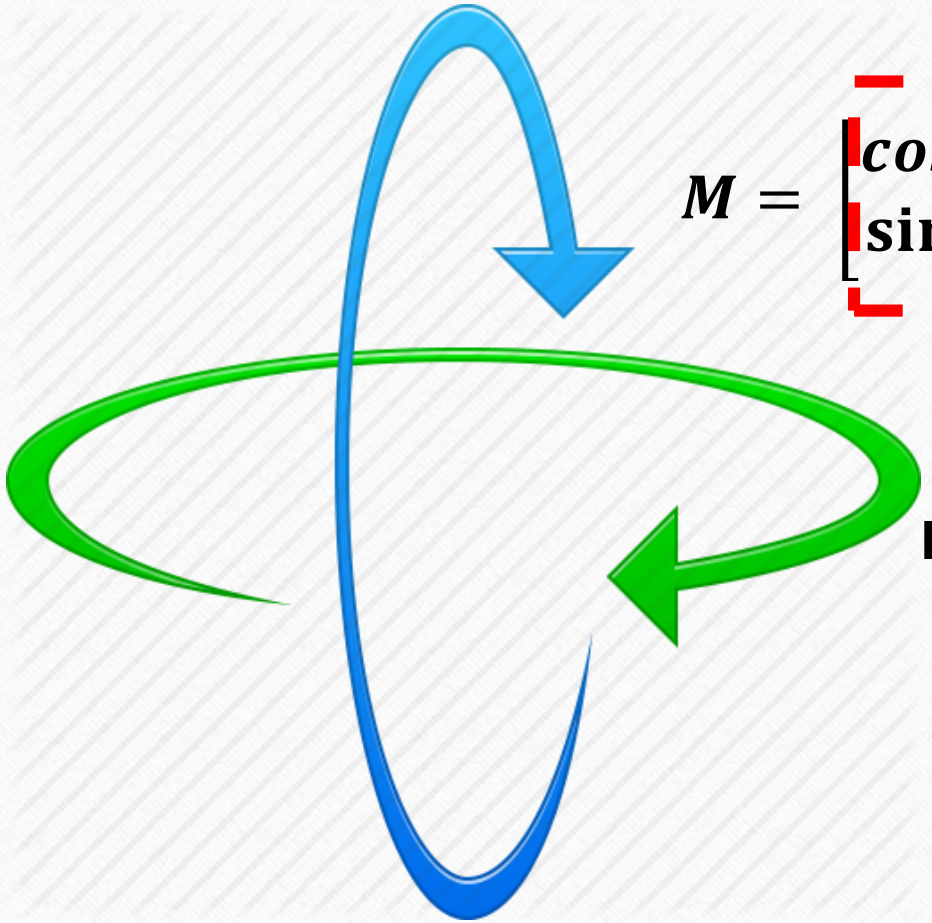
پیش پردازش داده ها

کتابخانه imutils

- نوشته Adrian Rosebrock
- شامل توابع ساده شده ای از توابع OpenCV
- و توابع اضافی که البته کاربردی است !

وبلاگ : pyimagesearch.com

آشنایی با ماتریس همگن دو بعدی



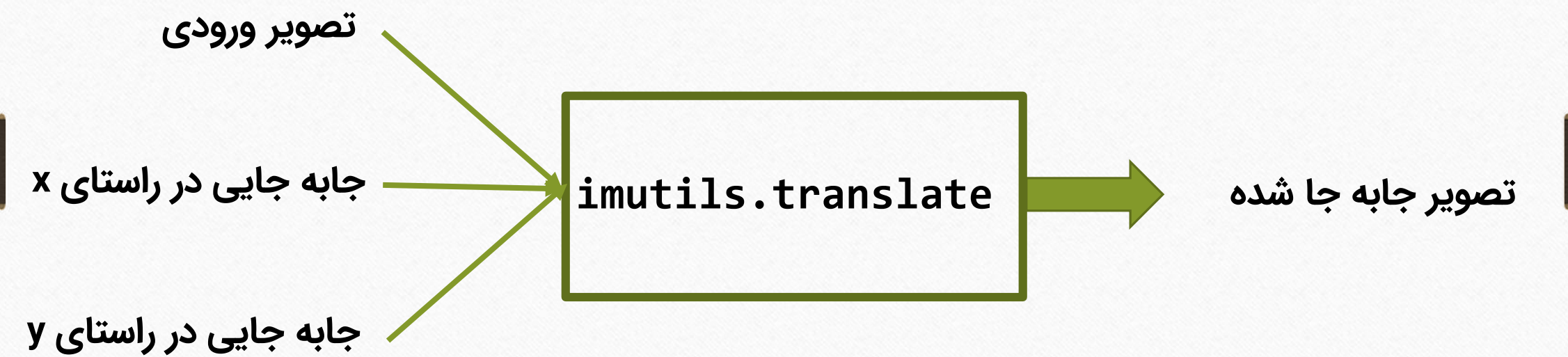
$$M = \begin{bmatrix} cos(\theta) & -sin(\theta) & t_x \\ \sin(\theta) & cos(\theta) & t_y \end{bmatrix}$$

ماتریس Translation

ماتریس Rotation

دستور 45 : جابه جایی تصویر (Translation)

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

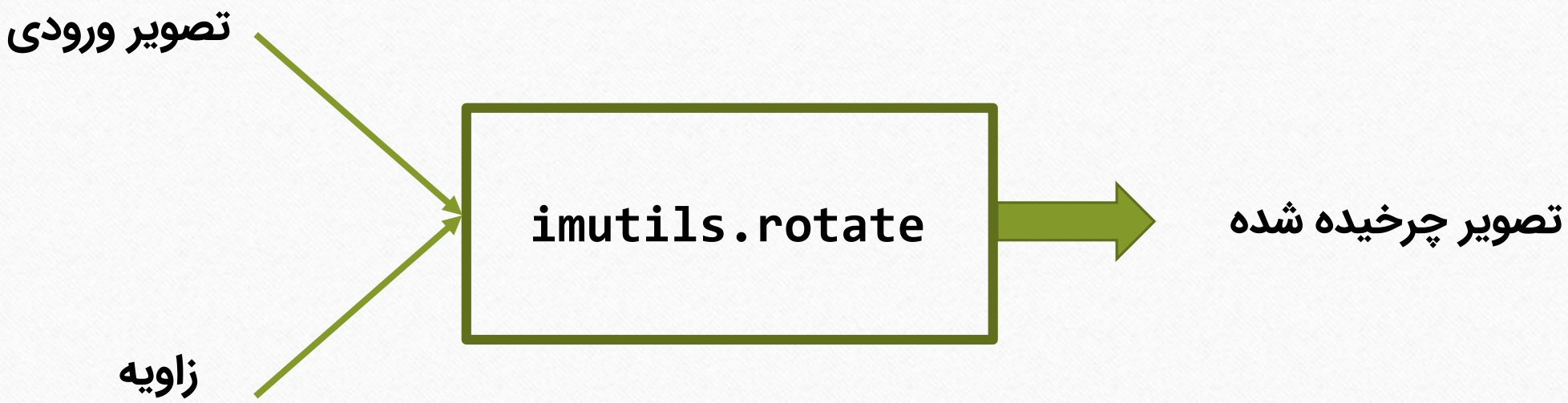


مثال:

```
translated = imutils.translate(img, 25, -75)
```

دستور 46 : چرخش تصویر (Rotation)

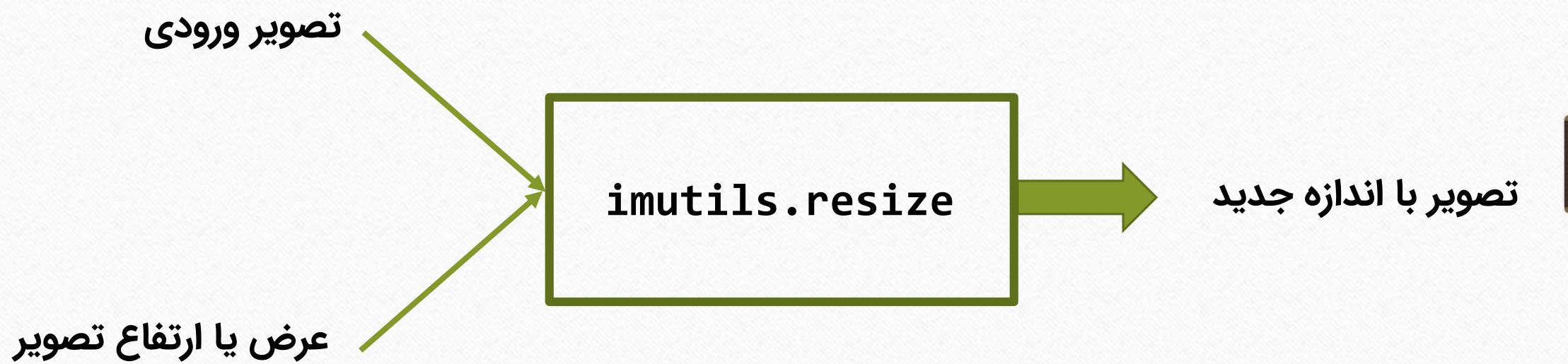
$$M = \begin{bmatrix} Cos(\theta) & -Sin(\theta) & 0 \\ Sin(\theta) & Cos(\theta) & 0 \end{bmatrix}$$



مثال:

```
rotated = imutils.rotate(img, angle= 90)
```

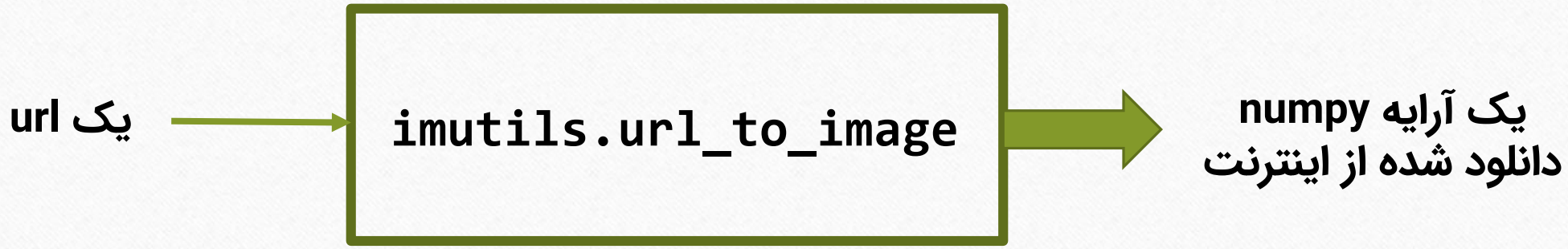
دستور 47 : تغییر اندازه تصویر



مثال:

```
rotated = imutils.resize(img, width = 400)
```

دستور 48 : خواندن عکس از صفحه وب

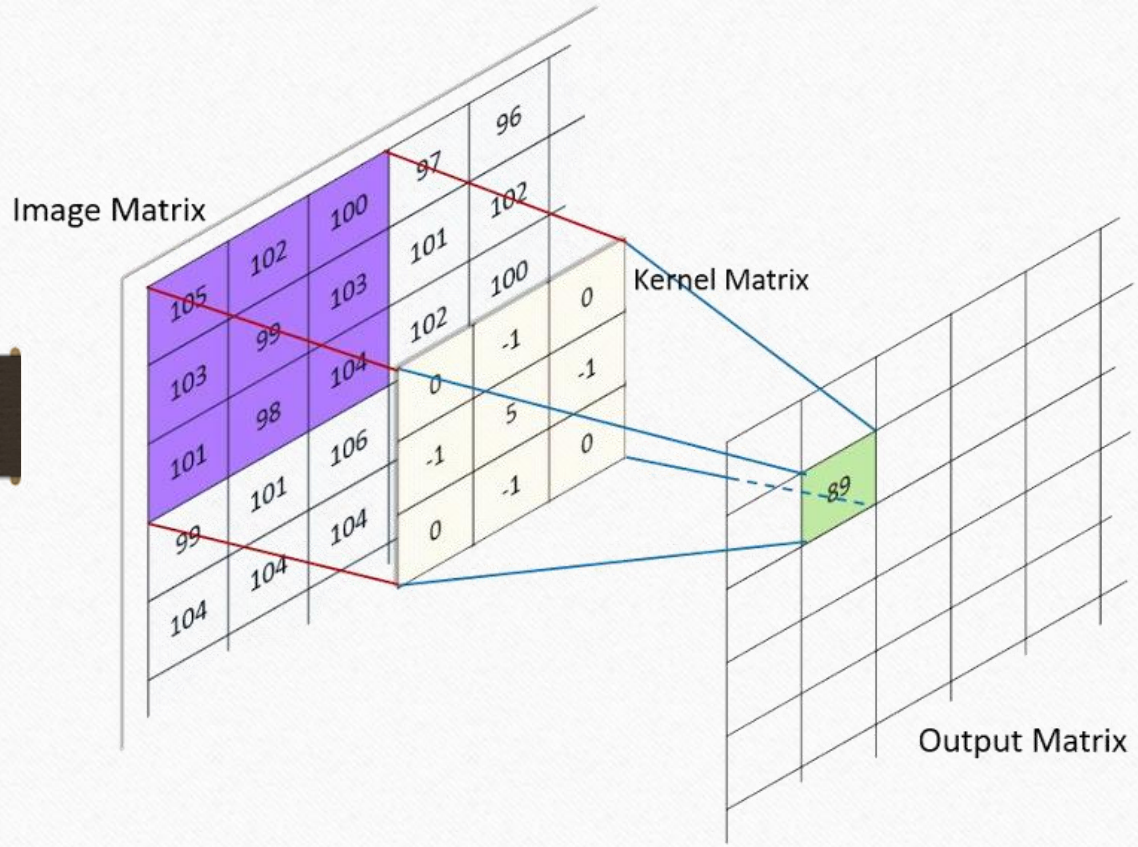


مثال:

```
Google_logo =imutils.url_to_image("https://www.google.com/logo.png")
```

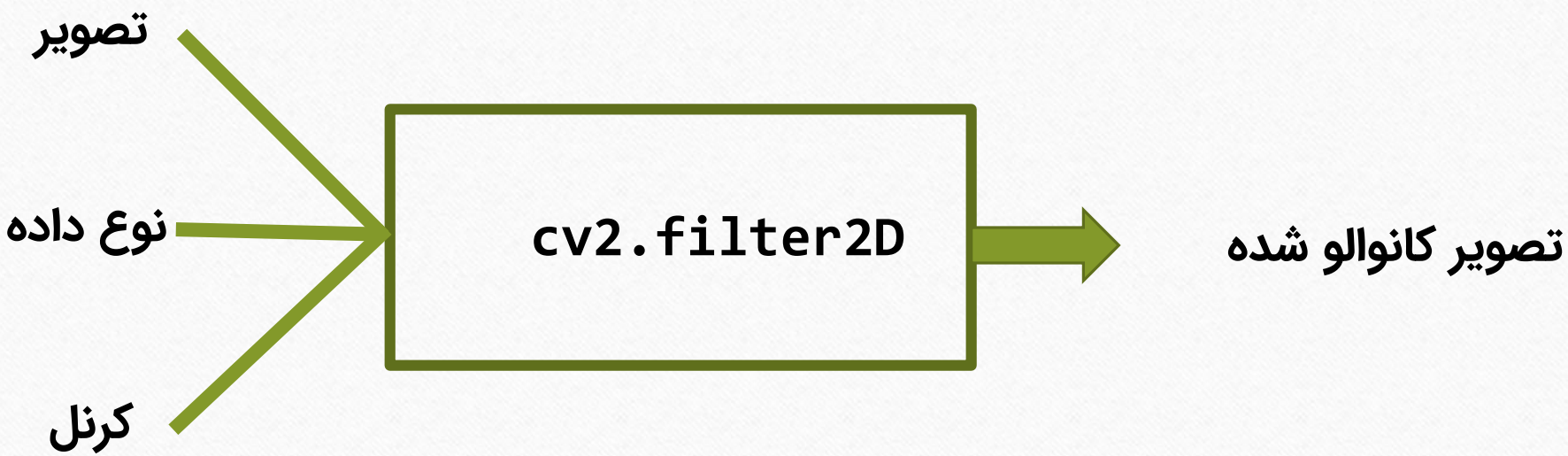
کانولوشن

کانولوشن و فیلتر



- ✓ ماتریس کرنل به صورت **افقی و عمودی** بر روی تصویر حرکت می کند و عمل کانولوشن را انجام می دهد.
- ✓ با کانولوشن **مقدار یک پیکسل با پیکسل های اطرافش کد می شود.**
- ✓ اندازه ماتریس کرنل همواره **عددی فرد** است.

دستور 49 : کانولوشن در OpenCV



مثال :

```
conv_img = cv2.filter2D(img, cv2.CV_8U, kernel)
```

- کرنل یک ماتریس است که میتوانیم آن را با کتابخانه numpy بسازیم.

دستور 50 : دستور blur

مثال :

```
blur_img = cv2.blur(img,(3,3))
```

دستور 51 : دستور Gaussian blur

مثال :

```
Gb_img = cv2.GaussianBlur(img,(3,3), 1)
```

فرمول محاسبه کرنل گاوسین:

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right); 1 \leq i, j \leq (2k+1)$$

شخیصت هفته : Andrew NG

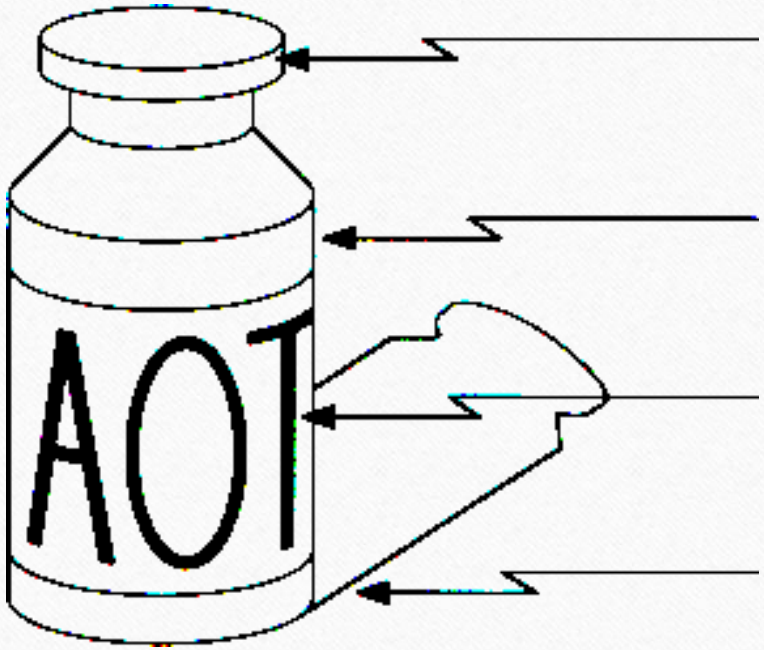
- یکی از سرمایه گذاران سیلیکون ولی
- استاد دانشگاه استنفورد
- موسس Baidu و Coursera و Google Brain

لبه یابی

لبه چیست ؟

لبه جایی است که تغییرات زیادی در intensity در یک همسایگی تصویر اتفاق می افتد.

عواملی که باعث ایجاد لبه می شوند:



تغییر شکل جسم در تصویر

تغییر در عمق تصویر

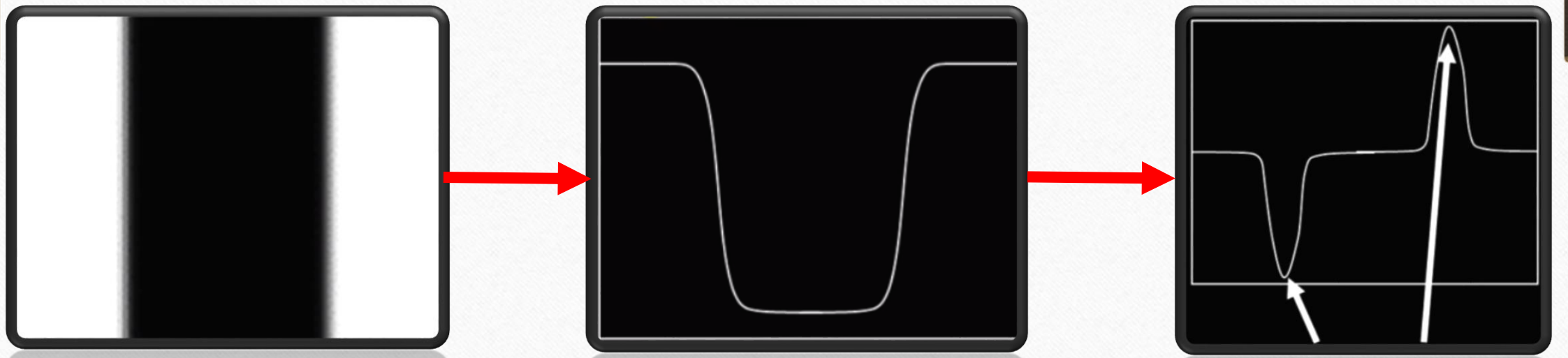
وجود رنگ های مختلف در یک جسم

حضور سایه های اجسام در تصویر

چرا لبه ها مهم هستند ؟

چگونه لبه ها را پیدا کنیم؟ ← لبه اختلاف در مقادیر بود. ← مشتق !

مشتق مرتبه اول | نمودار intensity یک ردیف | تصویر

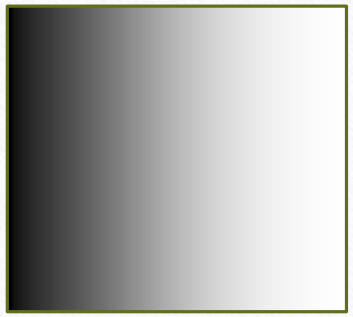


عملگر گرادیان (Gradient Operator)

ورودی تصویر و خروجی مشتق تصویر

این عملگر دارای دو مولفه در راستای x و y است.

$$\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$$



$$\nabla f = [\frac{\partial f}{\partial x}, 0]$$

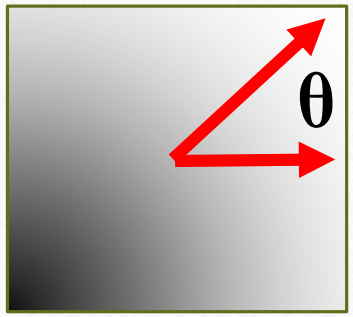
$$\nabla f = [0, \frac{\partial f}{\partial y}]$$

$$\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$$

جهت و اندازه گرادیان

جهت گرادیان:

$$\theta = \tan^{-1}\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right)$$

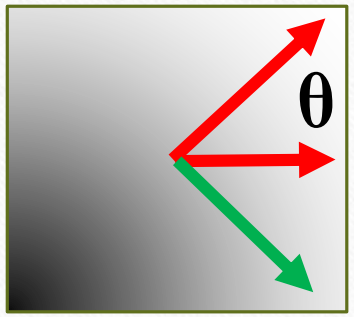


اندازه گرادیان:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

جهت لبه

عمود بر جهت گرادیان



مفهوم مشتق در یک تصویر:

مشتق پیوسته :

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon,\ y) - f(x,y)}{\varepsilon}$$

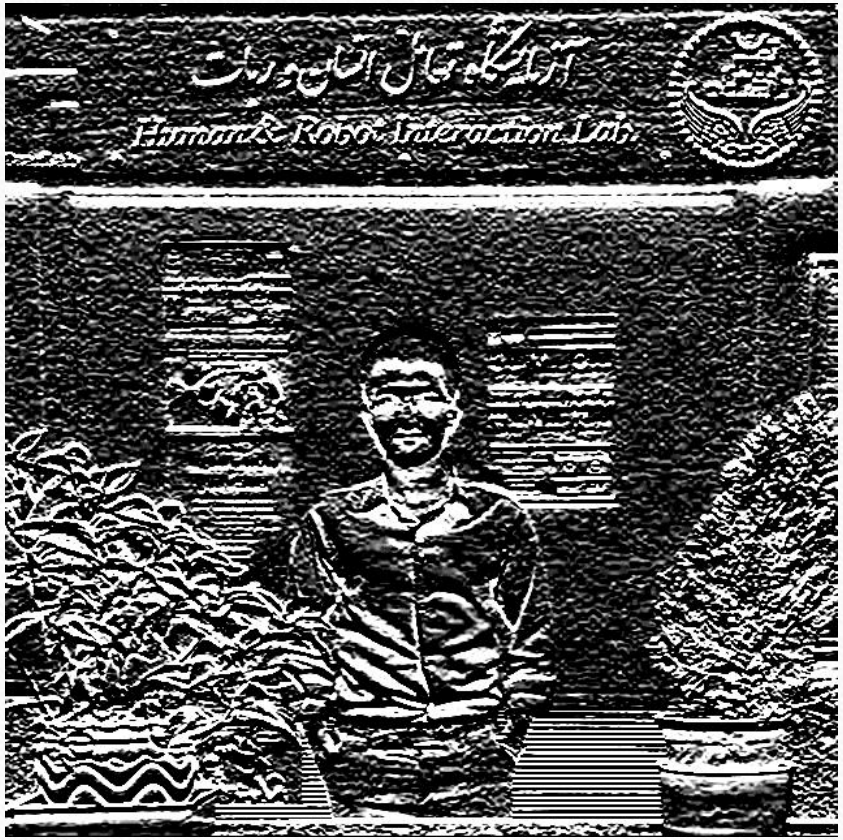
مشتق گسسته (در تصویر) :

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1} \approx f(x+1,\ y) - f(x,y)$$

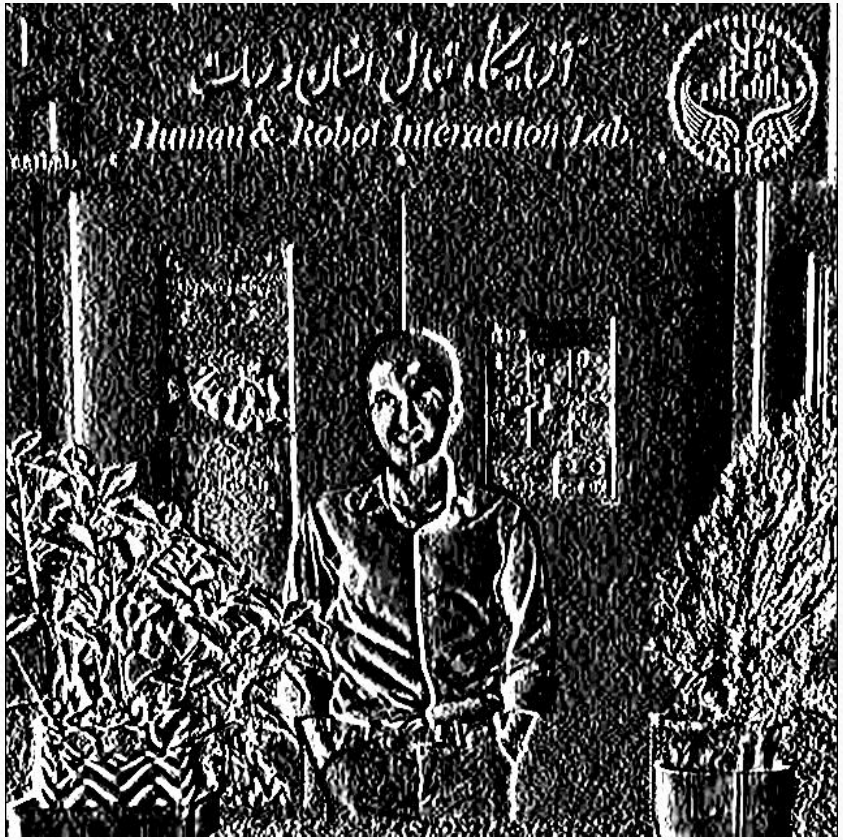
$$\frac{\partial f(x,y)}{\partial y} \approx \frac{f(x,y+1) - f(x,y)}{1} \approx f(x,\ y+1) - f(x,y)$$

مشتق در راستای X و Y

مشتق در راستای Y



مشتق در راستای X



چگونه این لبه ها را پیدا کنیم ؟

ایده : یک کرنل بدهیم و منتظر پاسخ بمانیم
سوال : چه کرنلی ؟

برخی از کرنل های معروف :

عملگر Prewitt

-1	0	+1
-1	0	+1
-1	0	+1

-1	0	+1
-1	0	+1
-1	0	+1

عملگر Sobel

-1	0	+1
-2	0	+2
-1	0	+1

-1	-2	-1
0	0	0
+1	+2	+1

آشنایی با عملگر Sobel

کرنل برای مشتق در جهت X

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

کرنل برای مشتق در جهت Y

$$\frac{1}{8}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

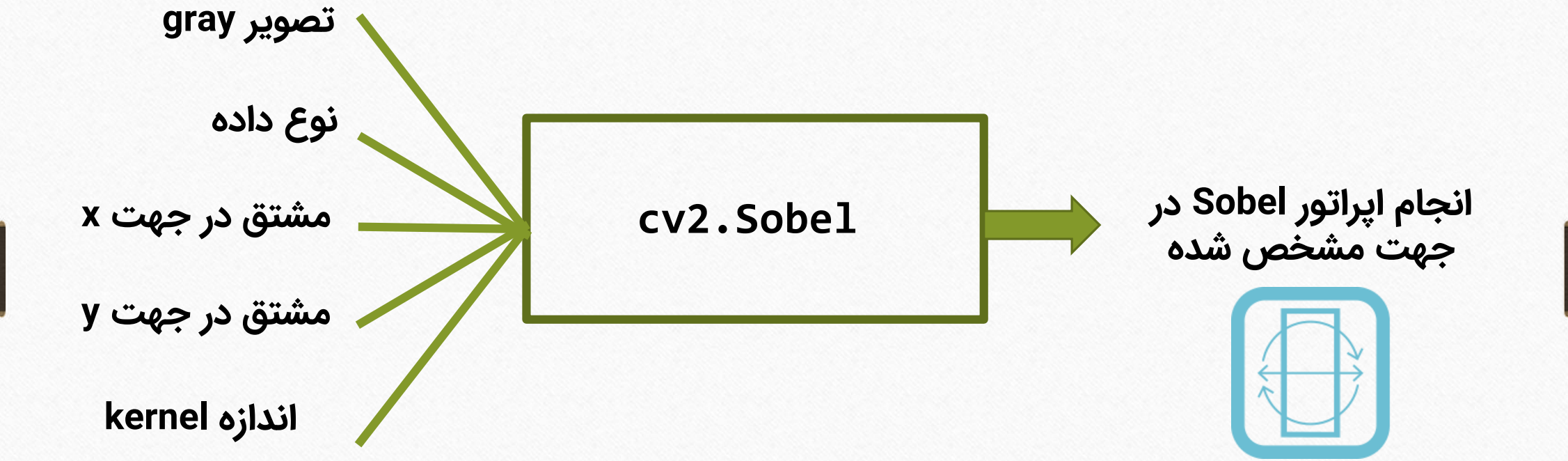
اندازه و جهت

$$\nabla I = [g_x \quad g_y]$$

$$\|\nabla f\| = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \tan^{-1}\left(g_x / g_y\right)$$

دستور Sobel : 32

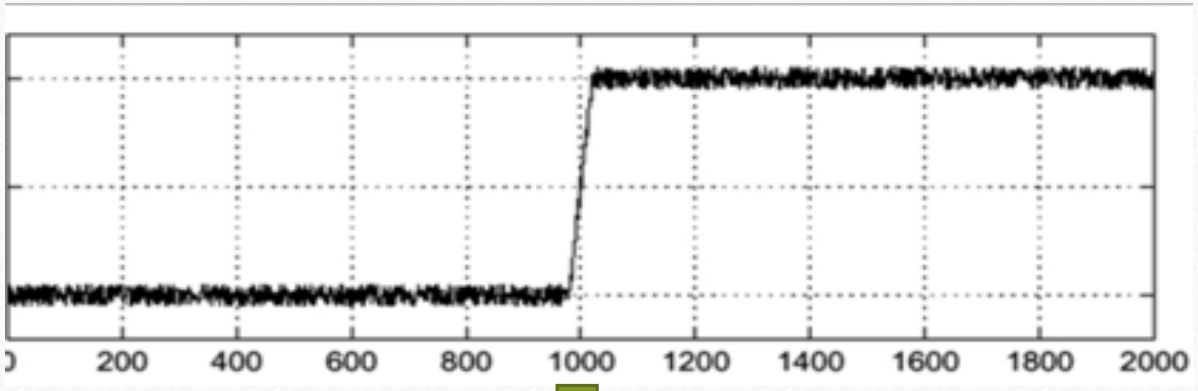


مثال :

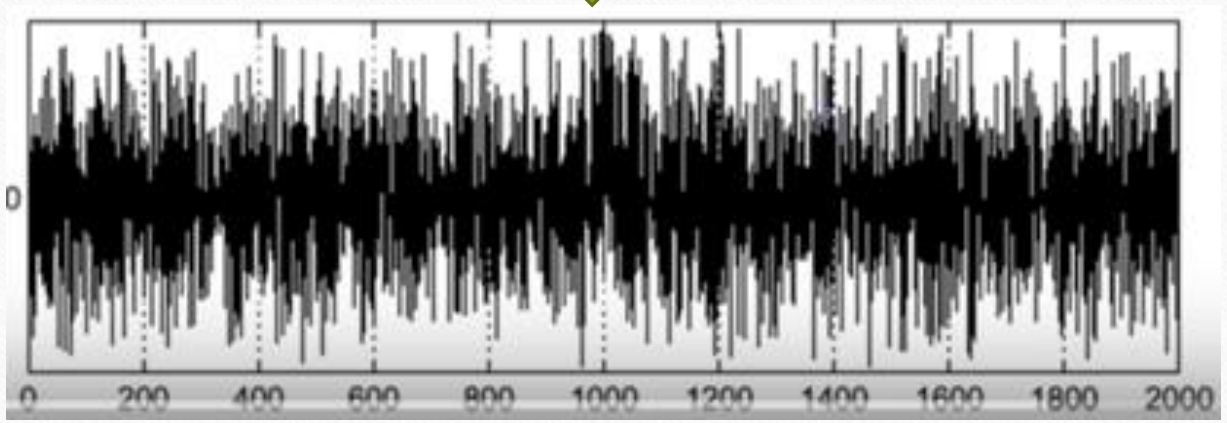
```
sobelX = cv2.Sobel(gray, cv2.CV_64F, dx=0, dy=1, ksize = 3)
```

و اما در دنیای واقعی !

$f(x)$

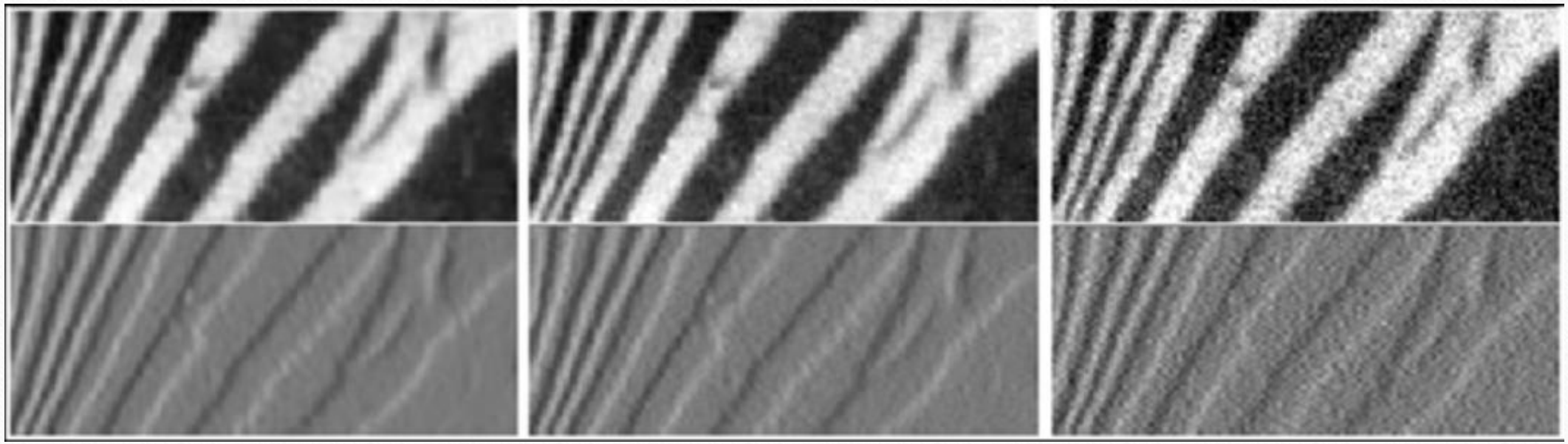


$\frac{d}{dx}(f(x))$



لبه کجاست ؟!

تاثیر حضور نویز بر لبه یابی



افزایش نویز گاوسی

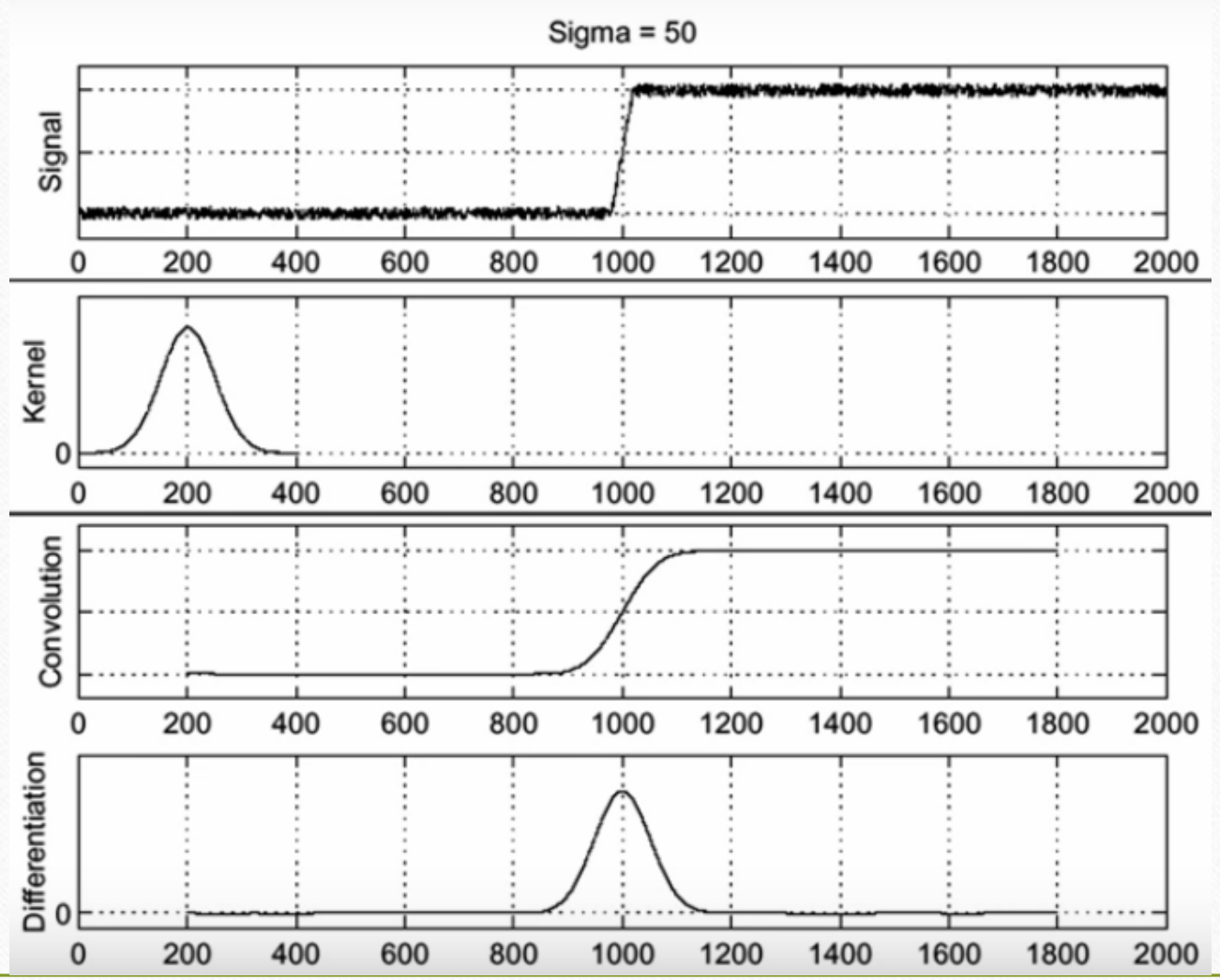
برای حل این مشکل چه کار کنیم ؟

f

h

$f * h$

$$\frac{\partial}{\partial x}(f * h)$$



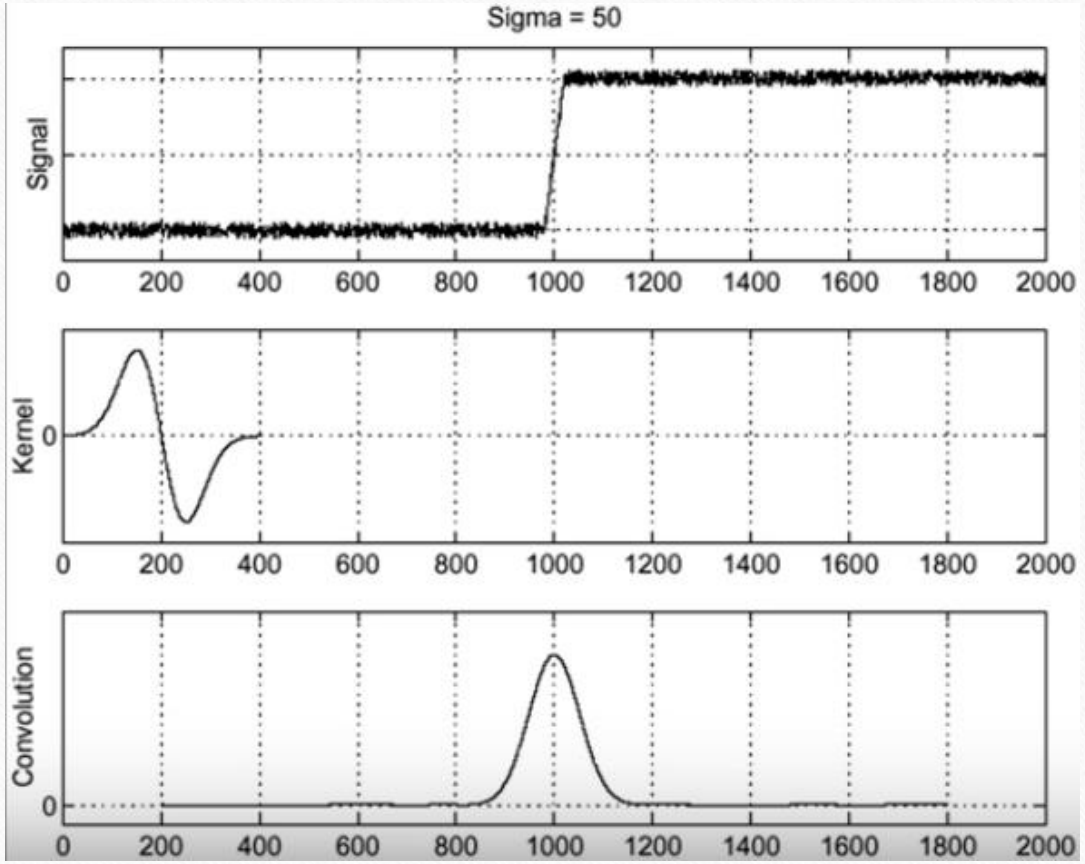
$$\frac{\partial}{\partial x}(h * f) = \frac{\partial}{\partial x}(h) * \mathbf{f}$$

میتوانیم از خواص ریاضی کانولوشن استفاده کنیم.

$$f$$

$$\frac{\partial}{\partial x}(h)$$

$$\frac{\partial}{\partial x}(f * h)$$



الگوریتم کنی برای لبه یابی

ارایه شده توسط جان کنی مهندس کامپیوتر استرالیایی

در سال 1986 و به عنوان پایان نامه ارایه شد

به صورت گسترده در بینایی ماشین ارایه می شود.

مراحل الگوریتم Canny

1. حذف نویز با فیلتر گاوسین
2. محاسبه گرادیان تصویر
3. Non Maximum Suppression
4. hysteresis thresholding

1 حذف نویز با فیلتر گاوسین

لبه یابی بسیار حساس به نویز است. به همین دلیل یک فیلتر گاوسین در ابتدا اعمال می شود.

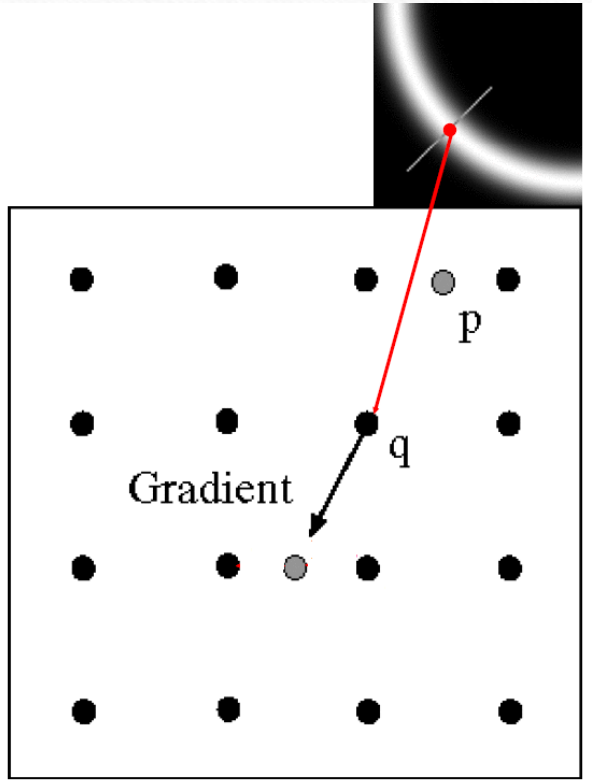
2 محاسبه گرادیان تصویر

به کمک عملگر Sobel گرادیان تصویر را در دو جهت x و y محاسبه می کنیم و magnitude را بدست می آوریم.

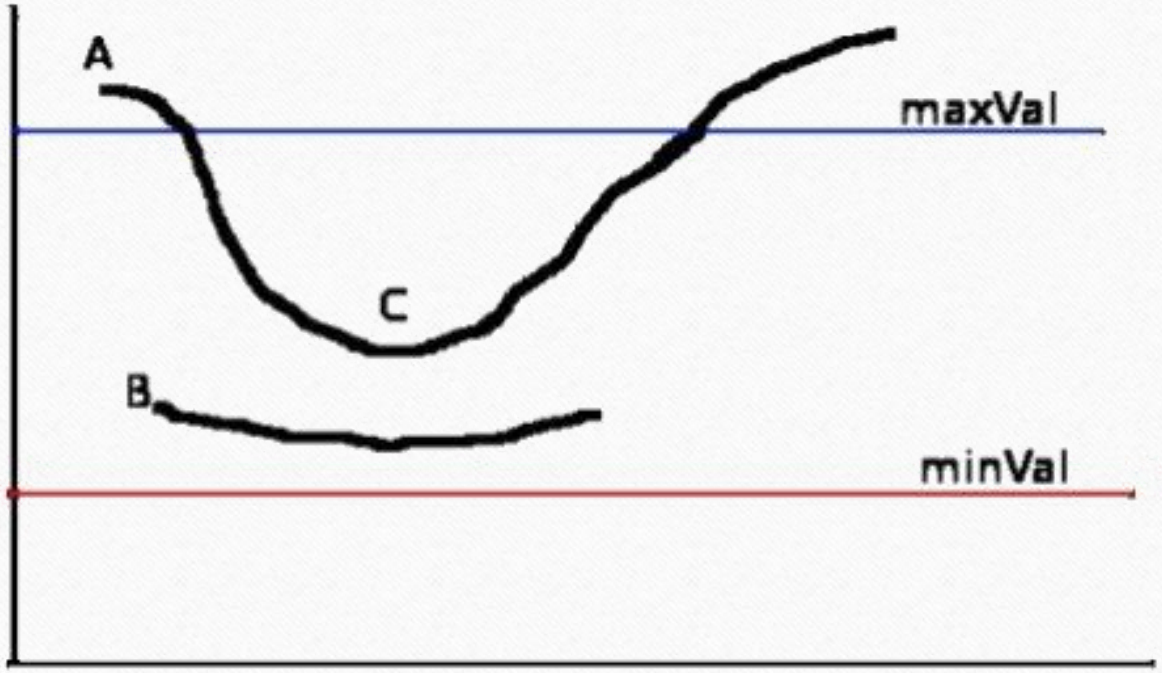
Non maximum supression

3

برای لبه ای که چند پیکسل نماینده آن هستند، باید پیکسلی با بیشترین Magnitude را نگه داشت و بقیه را حذف کرد.

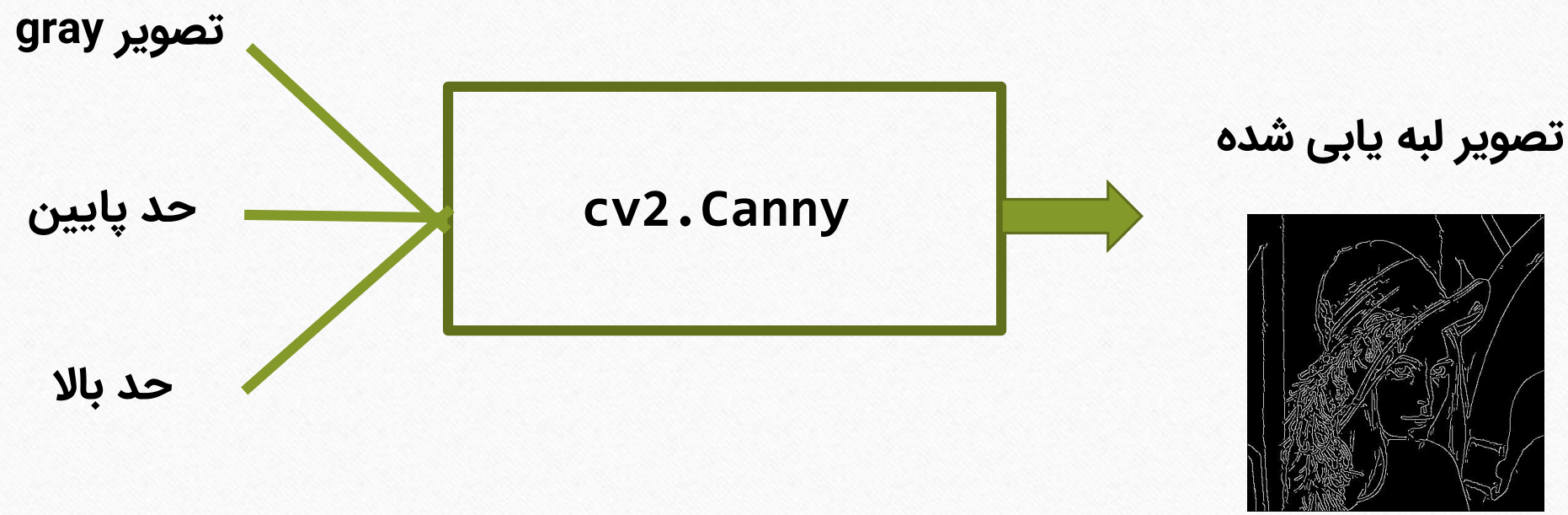


4 معرفی hysteresis thresholding



- نقطه A چون مقدار گرادیانش از مقدار maximum بیشتر است. پس لبه است.
- نقطه B مقدارش بین minVal و maxVal می باشد ولی چون به لبه ای قوی متصل نشده است ، لبه در نظر گرفته نمی شود.
- نقطه C مقدارش بین minVal و maxVal می باشد ولی چون به لبه ای قوی متصل شده است ، لبه در نظر گرفته می شود.

دستور 52 : الگوریتم Canny



مثال :

```
Canny_img = cv2.Canny(gray, 100, 200)
```

تمرین : به کمک عملگر Sobel و آنچه تا کنون آموخته اید در تصویر زیر بارکد را پیدا کنید.