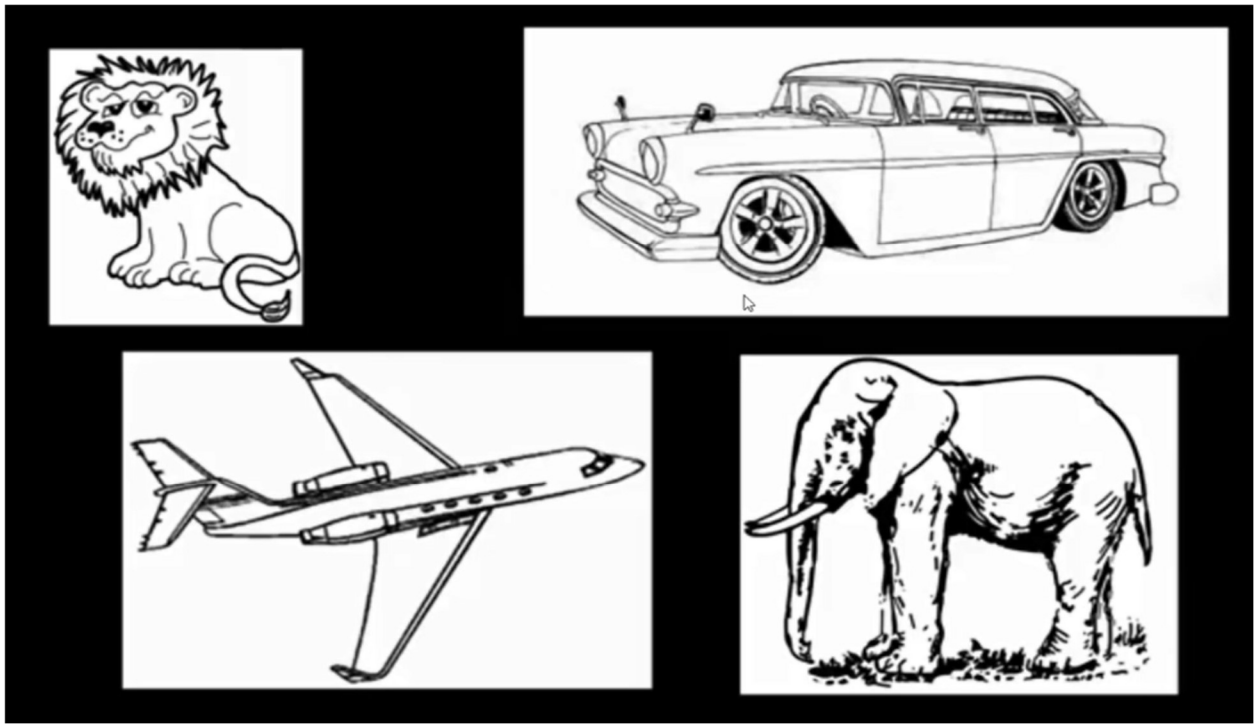
لبه در تصویر:

در پردازش تصویر، **لبه (Edge)** به نواحی‌ای گفته می‌شود که در آن‌ها تغییر ناگهانی و شدیدی در شدت روشنایی یا رنگ رخ می‌دهد. به بیان ساده‌تر، لبه‌ها همان مرزها یا خطوطی هستند که اشیاء مختلف را از یکدیگر جدا می‌کنند. به عنوان مثال، در یک تصویر از یک فنجان روی میز، ناحیه‌ای که شدت روشنایی یا رنگ از سطح سفید فنجان به پس‌زمینه‌ی تیره‌ی میز تغییر می‌کند، یک لبه محسوب می‌شود.

لبه‌ها حاوی بخش عمده‌ای از اطلاعات مهم در تصویر هستند، زیرا ساختار، شکل و مرز اجسام را نشان می‌دهند. بسیاری از الگوریتم‌های پردازش تصویر مانند **بخش‌بندی(Segmentation)** ، **تشخیص اشیاء** یا **ردیابی حرکت**، بر پایه‌ی شناسایی لبه‌ها عمل می‌کنند. در واقع اگر پس‌زمینه‌ی یک تصویر یکنواخت باشد، بخش اعظم اطلاعات تصویری را لبه‌ها در خود جای می‌دهند.

شخیص لبه معمولاً با محاسبه‌ی میزان تغییرات شدت روشنایی در نقاط مختلف تصویر انجام می‌شود. اگر تغییرات تدریجی باشند، ناحیه یکنواخت است؛ اما اگر تغییرات سریع و زیاد باشند، آن ناحیه یک لبه است. در عمل، این کار با استفاده از فیلترهایی مانند **Sobel** ، **Prewitt**، **Canny** یا **Laplacian** صورت می‌گیرد که تغییرات شدت روشنایی (مشتق‌ها یا گرادیان‌ها) را اندازه‌گیری می‌کنند.



تعریف گرادیان تصویر

در پردازش تصویر، **گرادیان (Gradient)** یک کمیت برداری است که نشان می‌دهد شدت روشنایی تصویر در هر نقطه با چه سرعتی و در چه جهتی تغییر می‌کند. به زبان ساده، گرادیان نشان‌دهنده‌ی **میزان و جهت تغییرات روشنایی** در تصویر است.

اگر در یک نقطه از تصویر شدت روشنایی به‌صورت ناگهانی تغییر کند (مثلاً از سفید به سیاه برسد)، مقدار گرادیان در آن نقطه بزرگ خواهد بود. در مقابل، اگر شدت روشنایی تقریباً ثابت باشد (مثلاً در ناحیه‌ی آسمان یا دیوار یکنواخت)، گرادیان نزدیک به صفر است.

اهمیت گرادیان

گرادیان اساس بسیاری از الگوریتم‌های پردازش تصویر است، زیرا:

- جهت گرادیان نشان می‌دهد مرز یا لبه در چه جهتی قرار دارد.
- بزرگی گرادیان مشخص می‌کند تغییرات روشنایی چقدر شدید است (لبه‌ی قوی یا ضعیف).
- عملگرهای معروف **Sobel**، **Prewitt** و **Canny** برای تشخیص لبه‌ها، همگی از محاسبه‌ی گرادیان استفاده می‌کنند.

Sobel :

لگوریتم **Sobel** یکی از متداول‌ترین روش‌ها برای تشخیص لبه در پردازش تصویر است. ایده‌ی اصلی این الگوریتم بر پایه‌ی محاسبه‌ی **گرادیان تصویر** است، یعنی میزان تغییر شدت روشنایی در نقاط مختلف Sobel .این کار را با استفاده از دو فیلتر (کرنل) انجام می‌دهد که یکی تغییرات در جهت افقی (x) و دیگری تغییرات در جهت عمودی (y) را محاسبه می‌کند. به این ترتیب، هم بزرگی تغییرات و هم جهت آن‌ها مشخص می‌شود.

در عمل، فیلتر Sobel یک ماتریس کوچک (معمولاً ۳×۳) است که روی تصویر حرکت داده می‌شود و در هر نقطه شدت تغییرات را محاسبه می‌کند. کرنل مربوط به محور افقی تغییرات روشنایی بین چپ و راست را می‌سنجد، و کرنل مربوط به محور عمودی تغییرات بالا و پایین را اندازه‌گیری می‌کند. این دو نتیجه با هم ترکیب می‌شوند تا بزرگی گرادیان و جهت آن به دست آید. بزرگی گرادیان نشان می‌دهد لبه در آن نقطه چقدر قوی است و جهت گرادیان زاویه‌ی لبه را مشخص می‌کند.

مزیت الگوریتم Sobel در مقایسه با روش‌های ساده‌تر این است که علاوه بر محاسبه‌ی مشتق (یعنی تغییرات شدت روشنایی)، عمل **صاف‌سازی (Smoothing)** را هم به صورت هم‌زمان انجام می‌دهد. به همین دلیل، در برابر نویز مقاوم‌تر از برخی روش‌های

دیگر است. در نهایت، خروجی این الگوریتم معمولاً یک تصویر است که لبه‌ها به شکل خطوط روشن روی زمینه‌ای تیره نمایان می‌شوند و می‌توان آن را در مراحل بعدی پردازش، مانند بخش‌بندی یا تشخیص اشیاء، استفاده کرد.

در پردازش تصویر، هدف از استفاده از فیلتر Sobel **تشخیص لبه‌ها** است. لبه همان ناحیه‌ای است که در آن شدت روشنایی (یا رنگ) به‌طور ناگهانی تغییر می‌کند. از نظر ریاضی، **مشتق** دقیقاً ابزاری است که این تغییرات را اندازه‌گیری می‌کند:

- اگر شدت روشنایی یکنواخت باشد ⟶ مشتق نزدیک به صفر است.
- اگر تغییر ناگهانی رخ دهد (مثلاً از سیاه به سفید) ⟶ مشتق مقدار بزرگی پیدا می‌کند.

پس با گرفتن مشتق، می‌توانیم نقاطی را که تغییر شدید دارند شناسایی کنیم، یعنی همان لبه‌ها.

در تصویر دوبعدی، تغییرات ممکن است در راستای افقی یا عمودی رخ دهند. برای همین در Sobel دو مشتق محاسبه می‌شود:

- **dx** تغییرات در راستای x لبه‌های عمودی
- **dy** تغییرات در راستای y لبه‌های افقی

سپس این دو را ترکیب می‌کنیم تا بزرگی و جهت گرادیان به دست آید. این کار کمک می‌کند تا لبه‌ها با هر جهتی (نه فقط افقی یا عمودی) شناسایی شوند.

```
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=5)
```

- این خط لبه‌ها را در **جهت افقی (x)** پیدا می‌کند.
- پارامترهای مهم:
  - `cv2.CV_64F`: نوع داده خروجی، تا مقادیر منفی هم ذخیره شوند.
  - `1, 0`: یعنی مشتق در جهت x گرفته شود.(dx=1, dy=0)
  - `ksize=5`: اندازه‌ی کرنل (هر چه بزرگ‌تر باشد لبه‌ها نرم‌تر و کمتر به نویز حساس‌اند).
- نتیجه: تصویر `sobel_x` فقط تغییرات افقی (لبه‌های عمودی) را نشان می‌دهد.

image (src): تصویر ورودی. معمولاً خاکستری (یک‌کاناله) است تا تفسیر گرادیان ساده باشد؛ بااین‌حال اگر چندکاناله باشد، مشتق برای هر کانال جداگانه محاسبه می‌شود.

cv2.CV_64F (ddepth): عمقِ داده خروجی. چون مشتق می‌تواند منفی شود و دامنه آن از ۰..۲۵۵ فراتر رود، از نوع شناور ۶۴ بیتی استفاده شده تا هم **اشباع نشود** و هم **علامت منفی** حفظ گردد. (گزینه‌های رایج: CV_16S، CV_32F، CV_64F). برای نمایش معمولاً بعداً قدرمطلق/نرمال‌سازی می‌کنند.

1 (dx): مرتبه مشتق در راستای x. مقدار ۱ یعنی مشتق مرتبه اول در جهت افقی؛ در نتیجه **لبه‌هایی که مرزهای عمودی دارند** (تغییر روشنایی در راستای افقی) برجسته می‌شوند. می‌توان ۲ یا بیشتر هم داد (به‌ندرت لازم می‌شود).

0 (dy): مرتبه مشتق در راستای y. اینجا صفر است، یعنی در راستای عمودی مشتقی گرفته نمی‌شود. برای لبه‌های افقی، برعکس: dx=0, dy=1.

ksize=5: اندازه کرنل سوبل (فقط مقادیر ۱، ۳، ۵، ۷ مجازند). هرچه بزرگ‌تر باشد، **هموارسازی بیشتری** هم‌زمان با مشتق‌گیری انجام می‌شود و پاسخ به نویز کمتر اما لبه‌ها نرم‌تر می‌شود. ksize=1 تیزتر و پرنویزتر، ksize=5 یا 7 ملایم‌تر است.

```
final_sobel = cv2.add(sobel_x, sobel_y)
```

دستور final_sobel = cv2.add(sobel_x, sobel_y) در کتابخانه OpenCV برای ترکیب دو تصویر یا دو ماتریس استفاده می‌شود. در اینجا منظور از دو تصویر، نتایج اعمال فیلتر سوبل در جهت‌های افقی و عمودی است. تفاوت اصلی این دستور با استفاده از عملگر جمع (+) در این است که cv2.add از روش **اشباع (saturation)** برای جمع کردن مقادیر پیکسل‌ها استفاده می‌کند؛ به این معنی که اگر حاصل جمع از 255 بیشتر شود، مقدار نهایی 255 باقی می‌ماند. در مقابل، استفاده از جمع معمولی در NumPy باعث ایجاد **overflow** می‌شود و مقادیر می‌توانند دوباره از صفر شروع شوند، که در پردازش تصویر نتیجه‌ی نادرستی ایجاد می‌کند.

دو متغیر sobel_x و sobel_y معمولاً خروجی فیلترهای سوبل در دو جهت X و Y هستند. فیلتر سوبل برای تشخیص لبه‌ها در تصویر به کار می‌رود؛ به‌طوری که sobel_x تغییرات شدت روشنایی در جهت افقی را محاسبه می‌کند (یعنی لبه‌های عمودی را برجسته می‌کند) و sobel_y تغییرات در جهت عمودی را محاسبه می‌کند (لبه‌های افقی را برجسته می‌کند). وقتی این دو تصویر را با هم جمع می‌کنیم، نتیجه نهایی شامل لبه‌های موجود در هر دو جهت است و به این ترتیب تصویری کامل‌تر از لبه‌های تصویر اصلی به دست می‌آید.

با این حال، جمع ساده تنها یکی از روش‌های ترکیب این دو گرادیان است و همیشه بهترین نتیجه را نمی‌دهد. روش بهتر، محاسبه بزرگی گرادیان با استفاده از فرمول برداری است که می‌تواند با دستور `cv2.magnitude(sobel_x, sobel_y)` انجام شود. این روش بزرگی بردار گرادیان را در هر پیکسل محاسبه می‌کند و در نتیجه شدت واقعی لبه‌ها را بهتر نشان می‌دهد. بنابراین، در حالی که `cv2.add` روشی سریع و ساده برای ترکیب دو گرادیان است، در کاربردهای دقیق‌تر از روش محاسبه بزرگی گرادیان استفاده می‌شود.

```
image = cv2.imread('images/little_girl.jpg',0)

# Extract Sobel Edges
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=5)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=5)

sobel_x = np.abs(sobel_x)
sobel_y = np.abs(sobel_y)
final_sobel = cv2.add(sobel_x, sobel_y)


plt.figure(figsize=[12,7])
plt.subplot(231);plt.imshow(image, cmap='gray');plt.title("Original");
plt.subplot(234);plt.imshow(sobel_x, cmap='gray');plt.title("sobel X");
plt.subplot(235);plt.imshow(sobel_y, cmap='gray');plt.title("sobel Y");
        plt.subplot(236);plt.imshow(final_sobel, cmap='gray');plt.title("Final");
```

در این کد، اعمال `np.abs()` روی خروجی‌های `sobel_x` و `sobel_y` نقش بسیار مهمی دارد، زیرا به نحوه محاسبه و نمایش لبه‌ها مربوط می‌شود.

وقتی فیلتر سوبل روی تصویر اعمال می‌شود، اساس کار آن محاسبه مشتق در جهت افقی و عمودی است. مشتق می‌تواند **مقدار مثبت یا منفی** داشته باشد، چون تغییر شدت روشنایی می‌تواند به دو جهت باشد:

- اگر شدت روشنایی از چپ به راست افزایش یابد، مشتق مثبت است.
- اگر شدت روشنایی از چپ به راست کاهش یابد، مشتق منفی است.

به همین دلیل، خروجی خام سوبل شامل مقادیر منفی و مثبت است. اگر این مقادیر مستقیماً به تصویر تبدیل شوند، مقادیر منفی به عنوان سطوح تیره و مقادیر مثبت به عنوان سطوح روشن دیده می‌شوند، در حالی که آنچه برای تشخیص لبه اهمیت دارد **مقدار تغییرات** است نه جهت آن.

با اعمال `np.abs()` ، ما **قدر مطلق مشتق‌ها** را می‌گیریم تا همه مقادیر مثبت شوند. این کار باعث می‌شود لبه‌ها صرف‌نظر از جهت تغییرات روشنایی، شدت یکسانی داشته باشند. به عبارت دیگر، چه تغییر از تاریک به روشن باشد، چه برعکس، شدت لبه بر اساس اندازه تغییر محاسبه و نمایش داده می‌شود. بدون این مرحله، بخشی از لبه‌ها ممکن است بسیار تیره یا حتی نامشخص باشند، چون مقادیر منفی در نمایش تصویری مشکلی ایجاد می‌کنند.

از نظر کاربرد عملی، این مرحله ضروری است تا:

1. تصویر نهایی فقط **شدت لبه‌ها** را نشان دهد و وابسته به جهت نباشد.
2. هنگام ترکیب دو گرادیان X و Y ، مقادیر درست جمع شوند. اگر قدر مطلق نگیریم، ممکن است مثبت و منفی همدیگر را خنثی کنند و لبه محو شود.

در واقع، این تبدیل باعث می‌شود که تصویر خروجی بیانگر قدرت واقعی لبه‌ها باشد، نه جهت آن‌ها.

```
src = cv2.imread("images/little_girl.jpg")

src = cv2.GaussianBlur(src, (3, 3), 0)

gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)


grad_x = cv2.Sobel(gray, cv2.CV_16S, 1, 0, ksize=3)
grad_y = cv2.Sobel(gray, cv2.CV_16S, 0, 1, ksize=3)


abs_grad_x = cv2.convertScaleAbs(grad_x)
abs_grad_y = cv2.convertScaleAbs(grad_y)

grad = cv2.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0)


plt.figure(figsize=[12,7])
plt.subplot(231);plt.imshow(src[...,::-1]);plt.title("Original");
plt.subplot(234);plt.imshow(abs_grad_x, cmap='gray');plt.title("grad x");
plt.subplot(235);plt.imshow(abs_grad_y, cmap='gray');plt.title("grad y");
plt.subplot(236);plt.imshow(grad, cmap='gray');plt.title("final grad");
```

در این کد ابتدا گرادیان تصویر خاکستری در دو جهت افقی و عمودی با استفاده از فیلتر سوبل محاسبه می‌شود. دستور `cv2.Sobel(gray, cv2.CV_16S, 1, 0, ksize=3)` گرادیان افقی را محاسبه می‌کند و `cv2.Sobel(gray, cv2.CV_16S, 0, 1, ksize=3)` گرادیان عمودی را. استفاده از نوع داده

`cv2.CV_16S` به دلیل آن است که مشتق می‌تواند مقادیر منفی تولید کند و نوع داده ۸ بیتی نمی‌تواند این مقادیر را ذخیره کند، در نتیجه از overflow جلوگیری می‌شود.

پس از محاسبه گرادیان‌ها، خروجی‌ها با دستور `cv2.convertScaleAbs()` به نوع داده ۸ بیتی و قدر مطلق تبدیل می‌شوند. این کار دو مزیت دارد: اول اینکه تمام مقادیر منفی به مثبت تبدیل می‌شوند تا شدت واقعی لبه‌ها بدون توجه به جهت تغییر روشنایی نمایش داده شود، و دوم اینکه تصویر آماده نمایش و پردازش‌های بعدی می‌شود، چون اکثر توابع OpenCV با تصاویر ۸ بیتی کار می‌کنند.

در نهایت، دو تصویر گرادیان افقی و عمودی با دستور `cv2.addWeighted()` ترکیب می‌شوند تا تصویر نهایی لبه‌ها ساخته شود. وزن هر گرادیان برابر ۰.۵ در نظر گرفته شده است تا ترکیب متعادل ایجاد شود و هم لبه‌های افقی و هم عمودی به یک اندازه در تصویر نهایی دیده شوند. این روش نسبت به جمع ساده گرادیان‌ها کنترل بهتری روی شدت لبه‌ها فراهم می‌کند و خروجی بصری واضح‌تری تولید می‌کند.

در پردازش تصویر، مخصوصاً در عملیات گرادیان و لبه‌یابی مثل فیلتر سوبل، انتخاب نوع داده اهمیت زیادی دارد، چون مشتق‌ها می‌توانند **مقادیر منفی و بزرگ‌تر از ۸ بیت** تولید کنند. در ادامه توضیح می‌دهم هر نوع داده چه کاربردی دارد و چرا معمولاً `cv2.CV_16S` یا `cv2.CV_64F` استفاده می‌شود:

1. **۸ بیتی بدون علامت**

(cv2.CV_8U)
این نوع داده دامنه‌ای از 0 تا 255 دارد و برای تصاویر معمولی مناسب است. اما وقتی مشتق گرفته می‌شود، ممکن است مقادیر منفی تولید شود. ۸ بیت بدون علامت نمی‌تواند عدد منفی ذخیره کند، بنابراین اگر از این نوع داده برای گرادیان استفاده کنیم، مقدار منفی به صفر یا مقدار اشتباه تبدیل می‌شود و لبه‌ها درست نمایش داده نمی‌شوند.

2. **۱۶ بیتی با علامت**

(cv2.CV_16S)
این نوع داده می‌تواند مقادیر مثبت و منفی بین -32768 تا 32767 را ذخیره کند. به همین دلیل برای گرادیان‌ها و سوبل ایده‌آل است، چون تغییرات شدت روشنایی هم مثبت و هم منفی هستند. بعد از محاسبه گرادیان، معمولاً با `cv2.convertScaleAbs()` به ۸ بیتی و قدر مطلق تبدیل می‌کنیم تا برای نمایش و ترکیب آماده باشد.

3. **۳۲ و ۶۴ بیتی اعشاری (cv2.CV_32F و**

cv2.CV_64F):

این نوع داده‌ها برای محاسبات دقیق و جلوگیری از overflow استفاده می‌شوند. وقتی تصویر بزرگ یا کرنل بزرگ داریم یا می‌خواهیم عملیات ریاضی پیچیده انجام دهیم، گرادیان می‌تواند مقادیر بسیار بزرگ یا کوچک تولید کند. با نوع داده‌های اعشاری، می‌توانیم این مقادیر را با دقت بالا نگه داریم و بعداً به دلخواه به نوع ۸ بیتی تبدیل کنیم.

به طور خلاصه:

- برای **تصاویر ورودی معمولی** از CV_8U استفاده می‌کنیم.
- برای **محاسبه گرادیان و مشتق‌ها** از CV_16S یا CV_64F استفاده می‌کنیم تا مقادیر منفی و بزرگ از بین نرود.
- در نهایت برای **نمایش تصویر لبه‌ها** به CV_8U تبدیل می‌کنیم (با قدر مطلق).

Laplacian :

فیلتر Laplacian بر اساس **مشتق مرتبه دوم** عمل می‌کند. در حالی که سوبل و گرادیان‌ها (مشتق اول) تغییرات شدت روشنایی را در یک جهت مشخص (افقی یا عمودی) اندازه‌گیری می‌کنند، Laplacian **تغییرات شدت روشنایی در همه جهات را همزمان** بررسی می‌کند. به زبان ساده، Laplacian به ما نشان می‌دهد که در کدام نقاط تصویر تغییرات شدت روشنایی خیلی سریع اتفاق می‌افتد و معمولاً این نقاط همان لبه‌ها هستند.

معمول‌ترین فرمول Laplacian در دو بعد به صورت ماتریسی است که پیکسل مرکزی منفی و پیکسل‌های همسایه مثبت یا منفی دارند. این فیلتر مقادیر منفی و مثبت تولید می‌کند و برای نمایش نهایی معمولاً قدر مطلق آن گرفته می‌شود.

در تصاویر دیجیتال، هر پیکسل به طور معمول یک عدد مثبت بین ۰ تا ۲۵۵ (برای ۸ بیت) دارد که نشان‌دهنده شدت روشنایی است. وقتی می‌گوییم «پیکسل مقدار منفی دارد» در واقع منظور **خروجی حاصل از فیلتر** است، نه مقدار اصلی پیکسل.

برای مثال، در **فیلتر Laplacian** مقدار پیکسل جدید بر اساس فرمول زیر محاسبه می‌شود:

پیکسل جدید=جمع وزن‌دار همسایه‌ها− (تعداد همسایه×مقدار پیکسل مرکزی)

- اگر شدت روشنایی پیکسل مرکزی بیشتر از میانگین همسایه‌ها باشد، نتیجه این محاسبه **منفی** خواهد بود.
- اگر پیکسل مرکزی کمتر از همسایه‌ها باشد، نتیجه مثبت خواهد شد.

به زبان ساده: مقدار منفی به این معنی است که **پیکسل مرکزی نسبت به اطرافش روشن‌تر است** و مقدار مثبت یعنی **پیکسل مرکزی نسبت به اطرافش تاریک‌تر است.**

چون تصاویر نمی‌توانند مقادیر منفی نمایش دهند، معمولاً بعد از اعمال فیلتر، **قدر مطلق** گرفته می‌شود تا شدت تغییر (لبه) بدون توجه به جهت آن نشان داده شود.

تفاوت‌های کلیدی باSobel

- **نوع مشتق:**
  - Sobel از **مشتق مرتبه اول** استفاده می‌کند و گرادیان را در یک جهت X یا Y محاسبه می‌کند.
  - Laplacian از **مشتق مرتبه دوم** استفاده می‌کند و تغییرات شدت روشنایی را در همه جهات بررسی می‌کند.
- **جهت‌گیری:**
  - Sobel جهت‌دار است، یعنی می‌توانیم لبه‌های افقی و عمودی را جداگانه ببینیم.
  - Laplacian جهت ندارد و تمام لبه‌ها را به صورت یک تصویر ترکیبی نمایش می‌دهد.
- **حساسیت به نویز:**
  - Laplacian نسبت به نویز حساس‌تر است، چون مشتق مرتبه دوم نوسانات کوچک را هم بزرگنمایی می‌کند. به همین دلیل معمولاً قبل از Laplacian از **Gaussian Blur**برای کاهش نویز استفاده می‌شود.
- **پیچیدگی محاسبات:**
  - Sobel ساده‌تر و کنترل‌پذیرتر است و اغلب در مراحل اولیه پردازش یا برای تشخیص جهت لبه استفاده می‌شود.
  - Laplacian سریع است و برای گرفتن تمام لبه‌ها یکجا مناسب است، اما جزئیات نویز را نیز تشدید می‌کند.

کاربردها

- Laplacianبرای **تشخیص لبه‌های دقیق و کلی** در یک تصویر استفاده می‌شود، مخصوصاً وقتی جهت لبه مهم نباشد.
- Sobelوقتی می‌خواهیم **لبه‌ها در یک جهت خاص** را استخراج کنیم یا گرادیان تصویر لازم باشد، مناسب‌تر است.

Canny:

لبه‌یابی **Canny** یکی از پیشرفته‌ترین و پرکاربردترین روش‌ها برای تشخیص لبه‌ها در پردازش تصویر است و هدف آن استخراج لبه‌های واضح و دقیق با حداقل نویز است. این الگوریتم شامل چند مرحله مهم است. ابتدا تصویر با یک **Gaussian Blur** محوشده می‌شود تا نویز و نوسانات کوچک روشنایی کاهش پیدا کند، زیرا گرادیان‌ها نسبت به نویز حساس هستند. سپس

گرادیان‌های افقی و عمودی محاسبه می‌شوند معمولاً با فیلتر Sobel تا **شدت و جهت تغییرات روشنایی** در هر پیکسل مشخص شود. این مرحله به ما می‌گوید کدام نقاط تصویر دارای تغییر سریع روشنایی هستند که احتمالاً لبه‌اند.

در مرحله بعد، **non-maximum suppression**اعمال می‌شود؛ این کار باعث می‌شود فقط پیکسل‌هایی که بیشترین شدت لبه در جهت گرادیان هستند نگه داشته شوند و پیکسل‌های اطراف که شدت کمتری دارند حذف شوند. این مرحله، لبه‌ها را نازک و دقیق می‌کند. سپس با استفاده از **double threshold**، لبه‌ها به دو دسته قوی و ضعیف تقسیم می‌شوند. پیکسل‌هایی که شدت بالاتر از آستانه بالایی دارند، لبه‌های قوی در نظر گرفته می‌شوند و پیکسل‌هایی که بین دو آستانه هستند، لبه‌های ضعیف محسوب می‌شوند.

در نهایت، مرحله **hysteresis** انجام می‌شود که لبه‌های ضعیف تنها زمانی نگه داشته می‌شوند که به لبه‌های قوی متصل باشند و بقیه حذف می‌شوند. این کار باعث می‌شود لبه‌های واقعی حفظ شوند و نویز یا نقاط پراکنده که شبیه لبه‌اند، حذف شوند. نتیجه نهایی تصویر لبه‌ای بسیار دقیق و تمیز است که لبه‌ها را با ضخامت یک پیکسل نمایش می‌دهد و جهت و شدت آن‌ها نیز در پردازش‌های بعدی قابل استفاده است. به همین دلیل الگوریتم Canny نسبت به فیلترهای ساده‌ای مثل Sobel و Laplacian از دقت و کیفیت بالاتری برخوردار است.

```
edges = cv2.Canny(blur, 100, 200)
```

ورودی اول: تصویر(`src`)

این ورودی همان تصویر اصلی است که می‌خواهیم روی آن لبه‌یابی انجام دهیم. معمولاً تصویر **سطح خاکستری (grayscale)** به تابع داده می‌شود، ولی اگر تصویر رنگی باشد، OpenCVخودش آن را تبدیل می‌کند. تصویر ورودی باید نوع داده‌ای قابل پردازش (مثل ۸ بیتی) داشته باشد.

ورودی دوم: آستانه پایین(`threshold1`)

این عدد تعیین می‌کند **چه پیکسل‌هایی با شدت تغییر کم به عنوان لبه ضعیف در نظر گرفته شوند** .پیکسل‌هایی که گرادیانشان کمتر از این مقدار است، نادیده گرفته می‌شوند و حذف می‌شوند. به زبان ساده، این مقدار برای فیلتر کردن تغییرات کوچک روشنایی یا نویز کاربرد دارد.

ورودی سوم: آستانه بالا(threshold2)

این عدد تعیین می‌کند **چه پیکسل‌هایی به عنوان لبه قوی شناخته شوند** .پیکسل‌هایی که گرادیانشان بالاتر از این مقدار است، به صورت قطعی لبه در نظر گرفته می‌شوند. همچنین پیکسل‌هایی که بین آستانه پایین و بالا هستند، فقط اگر به لبه‌های قوی متصل باشند نگه داشته می‌شوند این مرحله hysteresis نام دارد.