حد آستانه گذاری یا Thresholding:

مفهوم **حد آستانه گذاری (Thresholding)** در کتابخانه OpenCV یک تکنیک اساسی در پردازش تصویر است که برای **تبدیل یک تصویر خاکستری به یک تصویر باینری** (سیاه و سفید) استفاده می‌شود. هدف اصلی از حد آستانه گذاری، **جداسازی اشیاء مورد نظر (پیش‌زمینه) از پس‌زمینه** تصویر است.

به طور خلاصه، در این تکنیک، هر پیکسل در تصویر خاکستری با یک **مقدار آستانه (Threshold Value)** مقایسه می‌شود. بر اساس این مقایسه، مقدار پیکسل در تصویر خروجی (باینری) تعیین می‌گردد:

- اگر مقدار پیکسل **بزرگتر** از مقدار آستانه باشد، معمولاً به یک مقدار **حداکثر** (مثلاً 255 برای سفید) تنظیم می‌شود.
- اگر مقدار پیکسل **کوچکتر یا مساوی** با مقدار آستانه باشد، معمولاً به یک مقدار **حداقل** (مثلاً 0 برای سیاه) تنظیم می‌شود.

به این ترتیب، تصویر خاکستری به یک تصویر باینری تبدیل می‌شود که در آن اشیاء مورد نظر به صورت نواحی سفید (یا سیاه) و پس‌زمینه به صورت نواحی سیاه (یا سفید) ظاهر می‌شوند.

انواع اصلی حد آستانه گذاری در OpenCV:

OpenCVتوابع مختلفی برای انجام حد آستانه گذاری ارائه می‌دهد که هر کدام روش متفاوتی برای تعیین مقدار پیکسل خروجی بر اساس مقایسه با مقدار آستانه دارند. مهم‌ترین این توابع و انواع آن‌ها عبارتند از:

1. **حد آستانه گذاری ساده (Simple Thresholding):** در این روش، یک مقدار آستانه **سراسری** برای کل تصویر تعیین می‌شود .پنج نوع حد آستانه گذاری ساده را ارائه می‌دهد که در نحوه تعیین مقدار پیکسل خروجی متفاوت هستند:
   - `cv2.THRESH_BINARY`: اگر مقدار پیکسل بیشتر از آستانه باشد، به `maxVal`و در غیر این صورت به 0 تنظیم می‌شود.
   - `cv2.THRESH_BINARY_INV`: معکوس `cv2.THRESH_BINARY`.
   - `cv2.THRESH_TRUNC`: اگر مقدار پیکسل بیشتر از آستانه باشد، به مقدار آستانه و در غیر این صورت بدون تغییر باقی می‌ماند.
   - `cv2.THRESH_TOZERO`: اگر مقدار پیکسل بیشتر از آستانه باشد، بدون تغییر باقی می‌ماند و در غیر این صورت به 0 تنظیم می‌شود.
   - `cv2.THRESH_TOZERO_INV`: معکوس `cv2.THRESH_TOZERO`.

2. **حد آستانه گذاری تطبیقی :(Adaptive Thresholding)** در مواردی که نورپردازی در تصویر یکنواخت نیست، استفاده از یک مقدار آستانه سراسری ممکن است نتایج خوبی به دست ندهد. حد آستانه گذاری تطبیقی، مقدار آستانه را برای هر پیکسل بر اساس یک **ناحیه کوچک اطراف آن** محاسبه می‌کند. این روش برای تصاویری با نورپردازی متغیر بسیار مؤثر است. دو روش اصلی برای محاسبه آستانه تطبیقی وجود دارد:
   - `cv2.ADAPTIVE_THRESH_MEAN_C`: مقدار آستانه، میانگین مقادیر پیکسل در ناحیه همسایگی منهای یک مقدار ثابت `C`است.
   - `cv2.ADAPTIVE_THRESH_GAUSSIAN_C`: مقدار آستانه، میانگین وزنی (با استفاده از یک پنجره گوسی) مقادیر پیکسل در ناحیه همسایگی منهای یک مقدار ثابت `C`است.
3. **حد آستانه گذاری اوتسو :(Otsu's Thresholding)** این روش یک الگوریتم **خودکار** برای تعیین مقدار آستانه بهینه بر اساس هیستوگرام تصویر است. فرض بر این است که تصویر دارای دو ناحیه با شدت روشنایی متفاوت (بای‌مدال) است و هدف، یافتن آستانه‌ای است که واریانس درون کلاسی را به حداقل می‌رساند. برای استفاده از این روش، از فلگ `cv2.THRESH_OTSU`به همراه یکی از انواع حد آستانه گذاری ساده (معمولاً `cv2.THRESH_BINARY` یا `cv2.THRESH_BINARY_INV`) استفاده می‌شود.

کاربردهای حد آستانه گذاری:

حد آستانه گذاری یک تکنیک پیش‌پردازش مهم در بسیاری از کاربردهای بینایی ماشین است، از جمله:

- **تشخیص اشیاء :**جداسازی اشیاء مورد نظر از پس‌زمینه برای تحلیل بیشتر.
- **اسکن اسناد :**تبدیل تصاویر اسکن شده به تصاویر باینری برای تشخیص متن.(OCR)
- **تقسیم بندی تصویر :(Image Segmentation)** جداسازی نواحی مختلف در تصویر.
- **ردیابی اشیاء :**ساده‌سازی تصاویر برای ردیابی آسان‌تر اشیاء در ویدیو.
- **تحلیل تصاویر پزشکی :**برجسته کردن ساختارهای خاص در تصاویر پزشکی.

در OpenCV ، تابع اصلی برای انجام حد آستانه گذاری `cv2.threshold()` است که برای حد آستانه گذاری ساده و اوتسو استفاده می‌شود، و تابع `cv2.adaptiveThreshold()` برای حد آستانه گذاری تطبیقی به کار می‌رود.

در کتابخانه OpenCV ، از پردازش تصاویر باینری به دلایل متعددی استفاده می‌کنیم که مهم‌ترین آن‌ها عبارتند از:

1 **سادگی و سرعت پردازش :**تصاویر باینری تنها از دو مقدار پیکسل (معمولاً 0 و 255 یا سیاه و سفید) تشکیل شده‌اند. این سادگی باعث می‌شود که عملیات پردازشی روی آن‌ها بسیار **سریع‌تر و از نظر محاسباتی کم‌هزینه‌تر** از تصاویر خاکستری یا رنگی باشد. الگوریتم‌های زیادی وجود دارند که به طور خاص برای تصاویر باینری طراحی شده‌اند و عملکرد بسیار خوبی دارند.

2 **جداسازی آسان پیش‌زمینه از پس‌زمینه :**هدف اصلی از تبدیل تصاویر خاکستری به باینری از طریق حد آستانه گذاری، **جداسازی اشیاء مورد نظر (پیش‌زمینه) از پس‌زمینه** است. پس از باینری شدن تصویر، تشخیص و تحلیل اشیاء بسیار آسان‌تر می‌شود.

3 **استفاده در الگوریتم‌های تشخیص و ردیابی :**بسیاری از الگوریتم‌های تشخیص اشیاء (مانند تشخیص لبه، تشخیص گوشه، تشخیص کانتور) و ردیابی اشیاء به تصاویر باینری به عنوان ورودی نیاز دارند یا عملکرد بهتری روی آن‌ها دارند. تصاویر باینری اطلاعات غیرضروری را حذف کرده و بر روی ویژگی‌های مهم اشیاء تمرکز می‌کنند.

4 **کاربرد در پردازش مورفولوژیکی :عملیات مورفولوژیکی** (مانند فرسایش، انبساط، باز کردن، بستن) که برای حذف نویز، پر کردن حفره‌ها، جدا کردن اشیاء متصل و غیره استفاده می‌شوند، معمولاً بر روی **تصاویر باینری** اعمال می‌گردند.

5 **کاهش حجم داده :**تصاویر باینری به دلیل داشتن تنها دو مقدار پیکسل، می‌توانند به طور **بسیار کارآمدتری** نسبت به تصاویر خاکستری یا رنگی ذخیره شوند و حجم داده کمتری را اشغال کنند.

6 **ساده‌سازی تحلیل تصویر :**تحلیل تصاویر باینری به دلیل کاهش پیچیدگی داده‌ها بسیار ساده‌تر است. اندازه‌گیری ویژگی‌های اشیاء مانند مساحت، محیط، مرکز جرم و غیره به راحتی قابل انجام است.

7 **پیش‌پردازش برای OCR تشخیص نوری کاراکتر :**در سیستم‌هایOCR ، تبدیل تصاویر اسکن شده به تصاویر باینری یک مرحله پیش‌پردازش حیاتی است. این کار باعث می‌شود که کاراکترها به طور واضح از پس‌زمینه جدا شده و تشخیص آن‌ها برای الگوریتم‌های OCR آسان‌تر شود.

به طور خلاصه، استفاده از پردازش تصاویر باینری در OpenCV به ما امکان می‌دهد تا:

- **پردازش‌های سریع‌تر و کارآمدتری** انجام دهیم.
- **اشیاء مورد نظر را به راحتی از پس‌زمینه جدا کنیم.**
- از تصاویر به عنوان ورودی برای **الگوریتم‌های تشخیص و ردیابی** استفاده کنیم.
- از **عملیات مورفولوژیکی** به طور مؤثر بهره ببریم.

- **حجم داده را کاهش دهیم.**
- **تحلیل تصاویر را ساده‌تر کنیم.**
- تصاویر را برای **سیستم‌های OCR** آماده کنیم.

کد اصلی برای آستانه گذاری در open cv :

```
ret,threshold(src,thresh,maxVal,type[    ,dst])
```

بیایید هر یک از آرگومان‌های این تابع را با جزئیات بررسی کنیم:

- **Src** این آرگومان **تصویر ورودی (source image)** است. این تصویر باید یک **تصویر تک‌کاناله (grayscale)** باشد. اگر یک تصویر رنگی به این تابع بدهید، با خطا مواجه خواهید شد. برای استفاده از یک تصویر رنگی، ابتدا باید آن را با استفاده از توابعی مانند `cv2.cvtColor()` به فضای رنگی خاکستری تبدیل کنید.
- **Thresh** این آرگومان **مقدار آستانه (threshold value)** است. این یک مقدار عددی است که برای مقایسه با مقدار شدت روشنایی هر پیکسل در تصویر ورودی استفاده می‌شود. بر اساس این مقایسه و نوع آستانه گذاری انتخاب شده، مقدار پیکسل خروجی تعیین می‌گردد.
- **maxVal** این آرگومان **مقدار حداکثر (maximum value)** است. این مقداری است که به پیکسل‌هایی در تصویر خروجی اختصاص داده می‌شود که **مقدار شدت روشنایی آن‌ها در تصویر ورودی از مقدار آستانه (thresh) بیشتر باشد** .این مقدار معمولاً **255 یا سفید** است.
- **type** این آرگومان **نوع آستانه گذاری (thresholding type)** را مشخص می‌کند. این یک فلگ است که تعیین می‌کند چگونه مقدار پیکسل خروجی بر اساس مقایسه با مقدار آستانه تعیین شود.

تابع `cv2.threshold()` در OpenCV **دو مقدار** را برمی‌گرداند:

1. **مقدار آستانه استفاده شده(retval)**
2. **تصویر آستانه گذاری شده(dst)**

متغیری که شما در کد خود به عنوان `ret` نامگذاری کرده‌اید، برای **دریافت مقدار آستانه استفاده شده** از تابع `cv2.threshold()` به کار می‌رود.

دلایل اصلی برای بازگرداندن مقدار آستانه عبارتند از:

سازگاری با روش‌های آستانه گذاری خودکار مانند Otsu یکی از قدرتمندترین ویژگی‌های `cv2.threshold()` این است که می‌تواند به همراه فلگ `cv2.THRESH_OTSU` برای انجام **آستانه گذاری اوتسو** استفاده شود. الگوریتم اوتسو به طور خودکار مقدار آستانه بهینه را بر اساس هیستوگرام تصویر محاسبه می‌کند. در این حالت، مقدار بازگردانده شده در `ret`، **مقدار آستانه‌ای خواهد بود که توسط الگوریتم اوتسو تعیین شده است .**این مقدار ممکن است با مقدار اولیه‌ای که شما به عنوان آرگومان `thresh` داده‌اید متفاوت باشد در واقع، وقتی از `cv2.THRESH_OTSU` استفاده می‌کنید، معمولاً مقدار `thresh` را 0 قرار می‌دهید و مقدار آستانه واقعی در `ret` برگردانده می‌شود.