عملیات بیتی روی تصاویر:

عملیات Bitwise روی تصاویر، عملیاتی هستند که روی تک تک بیت‌های پیکسل‌های تصویر انجام می‌شوند. این عملیات در دسته‌بندی عملیات پردازش تصویر سطح پایین قرار می‌گیرند و کاربردهای فراوانی در زمینه‌های مختلف دارند.

عملگرهای Bitwise در OpenCV

OpenCV توابع مختلفی برای انجام عملیات Bitwise روی تصاویر ارائه می‌دهد. برخی از مهم‌ترین این توابع عبارتند از:

- **cv2.bitwise_and()**: این تابع عملگر AND منطقی را روی دو تصویر یا یک تصویر و یک اسکالر انجام می‌دهد.
- **cv2.bitwise_or()**: این تابع عملگر OR منطقی را روی دو تصویر یا یک تصویر و یک اسکالر انجام می‌دهد.
- **cv2.bitwise_xor()**: این تابع عملگر XOR منطقی را روی دو تصویر یا یک تصویر و یک اسکالر انجام می‌دهد.
- **cv2.bitwise_not()**: این تابع عملگر NOT منطقی را روی یک تصویر انجام می‌دهد.

کاربردهای عملیات Bitwise

عملیات Bitwise در OpenCV کاربردهای گوناگونی دارند که از جمله آن‌ها می‌توان به موارد زیر اشاره کرد:

- **ماسک کردن تصاویر :** با استفاده از عملگر AND می‌توان بخش‌های خاصی از تصویر را ماسک و بقیه قسمت‌ها را حذف کرد.
- **ترکیب تصاویر :** با استفاده از عملگرهای OR و XOR می‌توان تصاویر مختلف را با یکدیگر ترکیب کرد.
- **تشخیص لبه :** با استفاده از عملگر NOT و برخی دیگر از تکنیک‌های پردازش تصویر می‌توان لبه‌های اشیا را در تصویر تشخیص داد.
- **پردازش تصاویر باینری :** عملیات Bitwise به طور گسترده در پردازش تصاویری که فقط شامل دو رنگ سیاه و سفید هستند (تصاویر باینری) استفاده می‌شوند.

cv2.bitwise_and():

عملیات **Bitwise AND** به این معناست که دو عدد (یا دو پیکسل) را در سطح بیتی با هم مقایسه می‌کنیم؛ به طوری که **فقط وقتی خروجی 1 می‌شود که هر دو بیت ورودی برابر 1 باشند.** در مورد تصاویر، این عمل روی هر پیکسل و هر کانال رنگی جداگانه انجام می‌شود. تابع **`cv2.bitwise_and()`** با مقایسه‌ی بیتی پیکسل‌های متناظر در دو تصویر (یا یک تصویر و یک ماسک) عمل می‌کند. این تابع مقدار هر پیکسل در خروجی را فقط زمانی برابر با مقدار واقعی آن پیکسل قرار

می‌دهد که هر دو پیکسل ورودی در همان موقعیت، دارای مقدار غیرصفر (یا 1 در سطح بیتی) باشند؛ در غیر این صورت مقدار آن پیکسل در خروجی صفر (سیاه) می‌شود. به عبارت دیگر، این تابع بخش‌های مشترک بین دو تصویر را حفظ کرده و نواحی‌ای را که تنها در یکی از تصاویر وجود دارند، حذف می‌کند. اگر از پارامتر **mask** استفاده شود، عملیات AND تنها در نواحی‌ای که ماسک مقدار 255 دارد انجام می‌گیرد و سایر نقاط تصویر در خروجی سیاه می‌شوند. فرض کنید تصویری از یک منظره دارید و می‌خواهید فقط قسمت آسمان آن را نگه دارید و بقیه بخش‌ها را حذف کنید. برای این کار، ابتدا یک **ماسک باینری** ایجاد می‌کنید که در آن ناحیه‌ی آسمان سفید (مقدار 255) و سایر قسمت‌ها سیاه (مقدار 0) است. سپس با استفاده از دستور `cv2.bitwise_and(image, image, mask=mask)`، تابع فقط پیکسل‌های مربوط به آسمان را از تصویر اصلی انتخاب می‌کند و سایر قسمت‌ها را سیاه می‌نماید. نتیجه‌ی نهایی تصویری خواهد بود که در آن فقط آسمان قابل مشاهده است و بقیه‌ی بخش‌ها حذف شده‌اند؛ این همان عملکرد دقیق عملیات بیتی AND در سطح پیکسل‌هاست.

cv2.bitwise_or():

تابع **`cv2.bitwise_or()`** در کتابخانه‌ی OpenCV برای انجام عملیات منطقی **OR یا بیتی** بین دو تصویر یا بین یک تصویر و یک ماسک به کار می‌رود. این تابع مقدار هر پیکسل در خروجی را به‌گونه‌ای محاسبه می‌کند که اگر حداقل یکی از دو پیکسل ورودی در همان موقعیت مقدار غیرصفر (روشن) داشته باشد، مقدار خروجی نیز غیرصفر خواهد بود. به عبارت دیگر، نتیجه شامل تمام نواحی روشن از هر دو تصویر است و هر نقطه‌ای که در یکی از تصاویر دارای مقدار روشن باشد، در خروجی حفظ می‌شود. در صورتی که از پارامتر **mask** استفاده شود، عمل OR فقط در نقاطی انجام می‌گیرد که مقدار ماسک برابر 255 است و سایر نقاط تصویر خروجی سیاه می‌شوند. این تابع معمولاً برای ترکیب نواحی مختلف، ادغام چند ماسک باینری، یا نمایش تمامی بخش‌های فعال در دو تصویر استفاده می‌شود. به عنوان مثال، فرض کنید دو ماسک جداگانه از قسمت‌های مختلف یک تصویر دارید؛ با استفاده از `cv2.bitwise_or(mask1, mask2)` می‌توانید آن‌ها را ترکیب کنید تا هر دو ناحیه به‌صورت هم‌زمان در یک ماسک نهایی دیده شوند.

cv2.bitwise_xor():

تابع **`cv2.bitwise_xor()`** در کتابخانه‌ی OpenCV برای انجام عملیات منطقی **XOR یا «یا انحصاری» بیتی (** بین دو تصویر یا بین یک تصویر و یک ماسک استفاده می‌شود. این تابع مقدار هر پیکسل خروجی را طوری محاسبه می‌کند که تنها در صورتی غیرصفر (روشن) باشد که **دقیقاً یکی از دو پیکسل ورودی در همان موقعیت مقدار غیرصفر داشته باشد؛** اگر هر دو پیکسل صفر یا هر دو غیرصفر باشند، مقدار خروجی صفر (سیاه) خواهد بود. به بیان ساده‌تر، این تابع نواحی‌ای را مشخص می‌کند که در یکی از تصاویر وجود دارند اما در دیگری نیستند، و در نتیجه برای مقایسه یا جداسازی تفاوت‌های دو تصویر بسیار

مفید است. اگر از پارامتر **mask** استفاده شود، عمل XOR فقط در نواحی‌ای انجام می‌شود که مقدار ماسک برابر 255 است و بقیه‌ی نقاط تصویر خروجی صفر می‌گردند. برای مثال، اگر دو ماسک از قسمت‌های مختلف تصویر داشته باشید و بخواهید نواحی مشترک آن‌ها حذف شود تا فقط بخش‌های غیرمشترک باقی بمانند، می‌توانید از دستور `cv2.bitwise_xor(mask1, mask2)` استفاده کنید؛ در این حالت، خروجی فقط شامل قسمت‌هایی خواهد بود که در یکی از ماسک‌ها سفید هستند و در دیگری سیاه.

cv2.bitwise_not():

تابع **`cv2.bitwise_not()`** در کتابخانه‌ی OpenCV برای انجام عملیات منطقی **NOT** **نقیض یا مکمل بیتی** بر روی یک تصویر استفاده می‌شود. این تابع مقدار هر پیکسل تصویر را معکوس می‌کند؛ یعنی اگر مقدار پیکسل 0 (سیاه) باشد، آن را به 255 (سفید) تبدیل می‌کند و برعکس، اگر مقدار پیکسل 255 (سفید) باشد، آن را به 0 تغییر می‌دهد. در واقع، `cv2.bitwise_not()` تمام بیت‌های تصویر را برعکس می‌کند و نتیجه تصویری است که رنگ‌های آن کاملاً معکوس شده‌اند. این تابع برای ایجاد نگاتیو تصویر، تولید ماسک مکمل، یا جدا کردن پس‌زمینه و پیش‌زمینه در فرایندهای پردازش تصویر کاربرد دارد. به عنوان مثال، اگر ماسکی دارید که در آن ناحیه‌ی مورد نظر سفید و بقیه‌ی تصویر سیاه است، با اجرای دستور `cv2.bitwise_not(mask)` می‌توانید همان ماسک را به‌صورت معکوس به دست آورید، به طوری که ناحیه‌ی مورد نظر سیاه و بخش‌های دیگر سفید شوند.