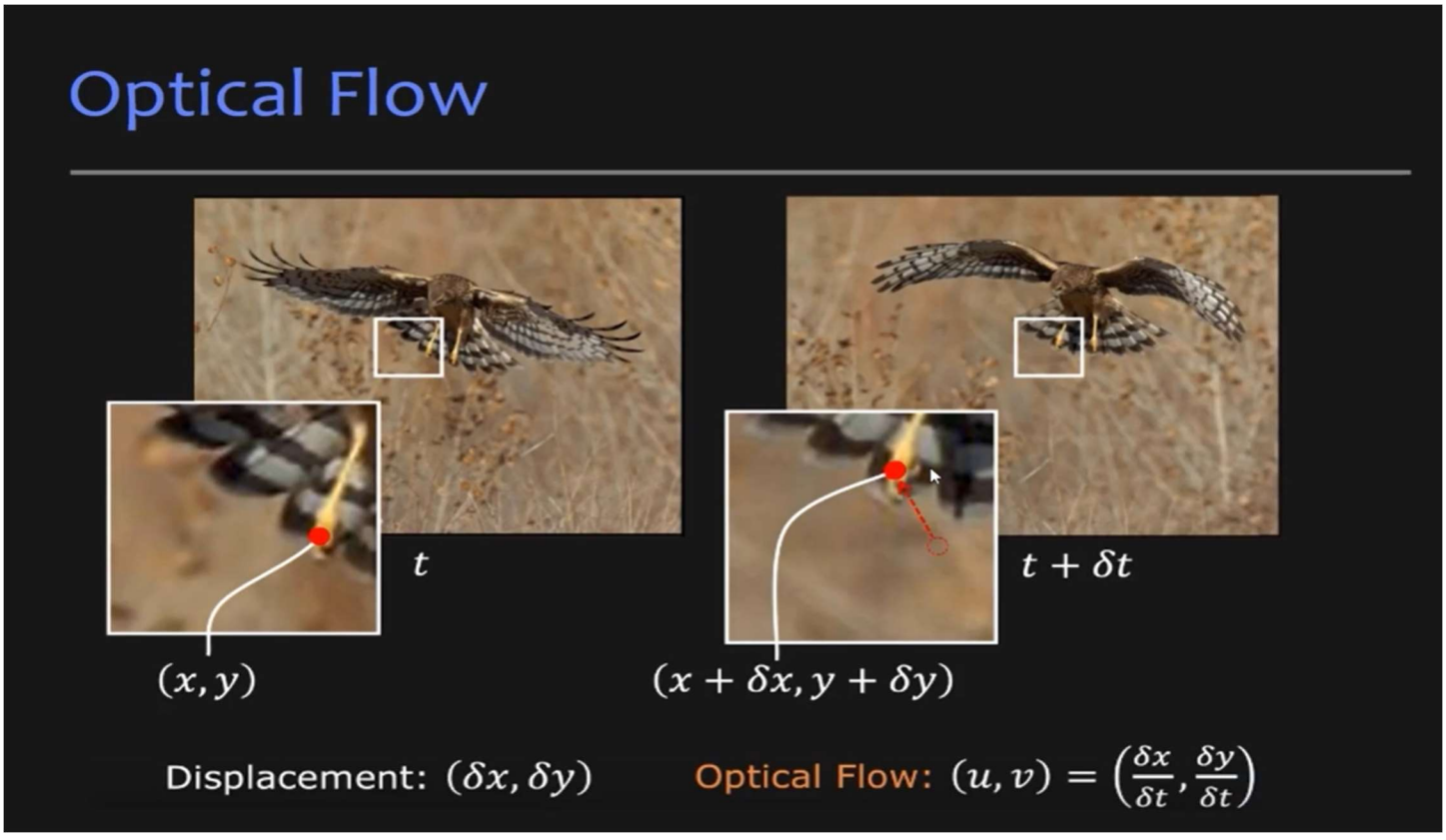
Optical Flow:

اپتیکال فلو (Optical Flow) در واقع روشی است برای تشخیص و دنبال‌کردن حرکت اشیا، سطوح یا بافت‌ها در یک دنباله از فریم‌های ویدیو. ایده‌ی اصلی این است که اگر در دو تصویر متوالی یک جسم کمی جابه‌جا شود، الگوریتم بتواند این جابه‌جایی را تشخیص دهد و جهت و اندازه‌ی حرکت را محاسبه کند. این اطلاعات به‌صورت یک میدان برداری (flow field) نمایش داده می‌شود که هر بردار نشان‌دهنده‌ی حرکت پیکسل‌ها بین دو فریم است.

در کتابخانه‌ی **OpenCV** چندین روش برای محاسبه‌ی اپتیکال فلو وجود دارد. دو روش معروف آن، **Lucas-Kanade** و **Farneback** هستند. روش Lucas-Kanade بیشتر برای دنبال کردن نقاط کلیدی (features) در یک ویدیو استفاده می‌شود، چون سریع و سبک است. در مقابل، روش Farneback میدان اپتیکال فلو را برای تمام پیکسل‌ها محاسبه می‌کند و برای تحلیل دقیق‌تر حرکات مناسب‌تر است، هرچند محاسبات آن سنگین‌تر است.

اپتیکال فلو در بسیاری از کاربردهای بینایی ماشین استفاده می‌شود. به‌عنوان مثال، در **ردیابی حرکت افراد یا اشیا، تخمین سرعت وسایل نقلیه، تشخیص حرکات دست یا بدن**، و حتی در **فشرده‌سازی ویدیو** برای تخمین تغییرات بین فریم‌ها. در رباتیک و پهپادها نیز از آن برای درک حرکت نسبی ربات نسبت به محیط استفاده می‌شود.



در عمل، برای استفاده از اپتیکال فلو در OpenCV تنها به چند خط کد نیاز است. مثلاً با تابع `cv2.calcOpticalFlowFarneback()` می‌توان میدان حرکت را به‌دست آورد و با توابع ترسیمی مثل `cv2.line()` یا `cv2.arrowedLine()` جهت حرکت را روی تصویر نمایش داد. در ردیابی نقاط خاص هم معمولاً از `cv2.calcOpticalFlowPyrLK()` استفاده می‌شود که بر پایه‌ی روش Lucas-Kanade کار می‌کند.

در نهایت، باید دانست که اپتیکال فلو همیشه دقیق نیست. عواملی مانند **تغییر نور**، **پنهان‌شدن اجسام**، **یا حرکات سریع** ممکن است باعث خطا شوند. برای بهبود نتایج، معمولاً از **پیش‌پردازش تصویر** (مثل فیلترگذاری یا هموارسازی)، یا ترکیب آن با روش‌های یادگیری ماشین و شبکه‌های عصبی استفاده می‌شود تا تخمین حرکت واقعی‌تر و پایدارتر گردد.

Sparse Optical Flow و Dense:

در اپتیکال فلو، دو رویکرد اصلی برای محاسبه‌ی حرکت وجود دارد **Sparse Optical Flow** و **Dense Optical Flow** تفاوت این دو در این است که چه تعداد از پیکسل‌های تصویر در فرآیند محاسبه‌ی حرکت مورد بررسی قرار می‌گیرند.

در **Sparse Optical Flow**، حرکت فقط برای **تعدادی از نقاط کلیدی** (مانند گوشه‌ها، لبه‌ها یا نقاطی با بافت واضح) محاسبه می‌شود. این نقاط معمولاً با روش‌هایی مثل **Shi-Tomasi** یا **Harris Corner Detection** انتخاب می‌شوند، چون در آن نواحی تغییرات روشنایی واضح‌تر و قابل‌اعتمادتر است. نتیجه این روش، مجموعه‌ای از بردارهای حرکت برای تعداد کمی نقطه است. مزیت اصلی آن سرعت بالا و کارایی مناسب در زمان واقعی (real-time) است، به همین دلیل در کاربردهایی مانند **ردیابی اشیا یا نقاط خاص** استفاده می‌شود.

در مقابل، **Dense Optical Flow** حرکت را برای **تمام پیکسل‌های تصویر** تخمین می‌زند. در این روش، هر پیکسل یک بردار حرکت دارد، بنابراین نتیجه یک "میدان برداری پیوسته" از جریان حرکت در سراسر تصویر است. این روش اطلاعات بسیار دقیق‌تری می‌دهد، اما محاسبات آن بسیار سنگین‌تر است. الگوریتم‌هایی مانند **Farneback** یا **DeepFlow** در OpenCV از این نوع هستند و معمولاً برای تحلیل‌های پیچیده‌تر یا یادگیری ماشین به کار می‌روند.

Lucas-Kanade:

روش **Lucas-Kanade** یکی از معروف‌ترین و پرکاربردترین الگوریتم‌ها برای محاسبه‌ی **Optical Flow** است که در سال ۱۹۸۱ معرفی شد. ایده‌ی اصلی آن بر پایه‌ی این فرض ساده است که شدت روشنایی یک نقطه در تصویر، در فریم‌های متوالی ثابت می‌ماند؛ یعنی اگر جسمی حرکت کند، ظاهرش تغییر نمی‌کند، فقط مکانش عوض می‌شود. بر این اساس، الگوریتم سعی می‌کند با مقایسه‌ی دو فریم متوالی، میزان جابه‌جایی پیکسل‌ها را محاسبه کند.

برای این کار، Lucas-Kanade فرض می‌کند که حرکت در یک ناحیه‌ی کوچک از تصویر (مثلاً یک بلوک چند پیکسلی) تقریباً یکسان است. سپس با بررسی تغییرات شدت روشنایی در آن ناحیه، حرکت مشترک آن بلوک را به‌صورت یک بردار (دارای جهت و اندازه) به‌دست می‌آورد. این کار باعث می‌شود روش نسبتاً سریع و مقاوم به نویز باشد، چون به‌جای بررسی تک‌تک پیکسل‌ها، از اطلاعات چند پیکسل در کنار هم استفاده می‌کند.

در عمل، این روش معمولاً برای **ردیابی نقاط کلیدی** (Feature Tracking) در ویدیو استفاده می‌شود. در OpenCV ، این الگوریتم در تابع `cv2.calcOpticalFlowPyrLK()` پیاده‌سازی شده است. معمولاً ابتدا نقاط مهم تصویر با روش‌هایی مثل **Shi-Tomasi** یا **Harris Corner Detection** پیدا می‌شوند و سپس با Lucas-Kanade در فریم‌های بعدی دنبال می‌گردند. نتیجه‌ی آن، مسیر حرکت نقاط در طول زمان است که می‌تواند برای تشخیص حرکت اجسام، تخمین مسیر دوربین یا حتی تحلیل حرکات بدن انسان به کار رود.

روش **Lucas-Kanade** به‌صورت سنتی در دسته‌ی **Sparse Optical Flow** قرار می‌گیرد، چون فقط حرکت نقاط کلیدی را دنبال می‌کند، نه کل تصویر. با این حال، نسخه‌های پیشرفته‌تر آن (مثل *Pyramidal Lucas-Kanade*) می‌توانند در چندین مقیاس کار کنند و نتایج دقیق‌تری ارائه دهند، ولی همچنان اساس آن بر ردیابی نقاط منتخب است، نه تمام پیکسل‌ها.

corner detection:

در روش **Lucas-Kanade**، الگوریتم خودش به‌تنهایی نقاط خاصی از تصویر را برای دنبال‌کردن انتخاب نمی‌کند؛ بلکه برای عملکرد بهتر، باید نقاطی را به آن بدهیم که **به‌راحتی قابل تشخیص و دنبال‌کردن** باشند. این نقاط خاص معمولاً همان چیزی هستند که به آن‌ها **corner** (گوشه یا زاویه) گفته می‌شود، و فرآیند شناسایی آن‌ها را **corner detection** می‌نامند.

ایده‌ی اصلی corner detection این است که گوشه‌ها در تصویر، نواحی‌ای هستند که در دو جهت مختلف تغییرات روشنایی شدیدی دارند. به بیان ساده‌تر، اگر یک پیکسل در یک تصویر در میان دو لبه یا بافت متقاطع قرار داشته باشد، جابه‌جایی آن در هر جهت باعث تغییر قابل توجهی در شدت روشنایی می‌شود؛ بنابراین این نقاط برای ردیابی در فریم‌های بعدی بسیار مناسب‌اند، چون الگوریتم می‌تواند آن‌ها را به‌خوبی از پس‌زمینه یا نواحی یکنواخت تشخیص دهد.

در عمل، معمولاً از الگوریتم‌هایی مثل **Harris Corner Detector** یا **Shi-Tomasi Corner Detector** برای انتخاب این نقاط استفاده می‌شود. در OpenCV ، تابع `cv2.goodFeaturesToTrack()` برای شناسایی این نقاط به‌کار می‌رود که در واقع پیاده‌سازی روش Shi-Tomasi است .سپس خروجی این تابع به الگوریتم Lucas-Kanade داده می‌شود تا در فریم‌های بعدی موقعیت همین نقاط را دنبال کند. به همین دلیل می‌توان گفت corner detector در روش Lucas-Kanade مثل "چشم انتخاب‌گر" است که تصمیم می‌گیرد کدام نقاط ارزش دنبال‌کردن دارند.