Corner Detection:

تشخیص گوشه‌ها (Corner Detection) یکی از مهم‌ترین مراحل در پردازش تصویر و بینایی ماشین است. گوشه‌ها نقاطی هستند که تغییرات شدیدی در شدت روشنایی در دو جهت مختلف دارند. به دلیل همین ویژگی، این نقاط اطلاعات زیادی درباره ساختار و شکل اجسام در تصویر به ما می‌دهند و معمولا برای تشخیص ویژگی‌ها (Feature Extraction) یا تطبیق تصاویر (Image Matching)به کار می‌روند. در کتابخانه **OpenCV**، الگوریتم‌های متنوعی برای تشخیص گوشه‌ها پیاده‌سازی شده‌اند که می‌توانند در کاربردهای مختلف مانند ردیابی اشیا، بازسازی سه‌بعدی و تشخیص الگو استفاده شوند.

یکی از الگوریتم‌های پایه‌ای در این حوزه، **روش Harris Corner Detector** است. این روش با استفاده از ماتریس گرادیان تصویر (Gradient) تغییرات شدت روشنایی در اطراف هر پیکسل را محاسبه می‌کند و نقاطی را که تغییرات در هر دو جهت افقی و عمودی زیاد است به‌عنوان گوشه مشخص می‌کند. درOpenCV ، این الگوریتم با تابع `cv2.cornerHarris()` پیاده‌سازی شده و کاربر می‌تواند با تنظیم پارامترهایی مانند اندازه پنجره و مقدار حساسیت، نتایج را بهبود دهد.

روش دیگری که در OpenCV بسیار پرکاربرد است، **Shi-Tomasi Corner Detector**می‌باشد. این الگوریتم بهبود یافته‌ی روش Harris است و به جای استفاده از تابع پاسخHarris ، از کمترین مقدار ویژه (Eigenvalue) ماتریس گرادیان استفاده می‌کند. نتیجه این است که فقط نقاطی که از نظر پایداری بهتر هستند به عنوان گوشه انتخاب می‌شوند. درOpenCV ، این الگوریتم با تابع `cv2.goodFeaturesToTrack()`در دسترس است و برای تشخیص نقاط کلیدی در ردیابی ویژگی‌ها (Feature Tracking) بسیار مناسب است.

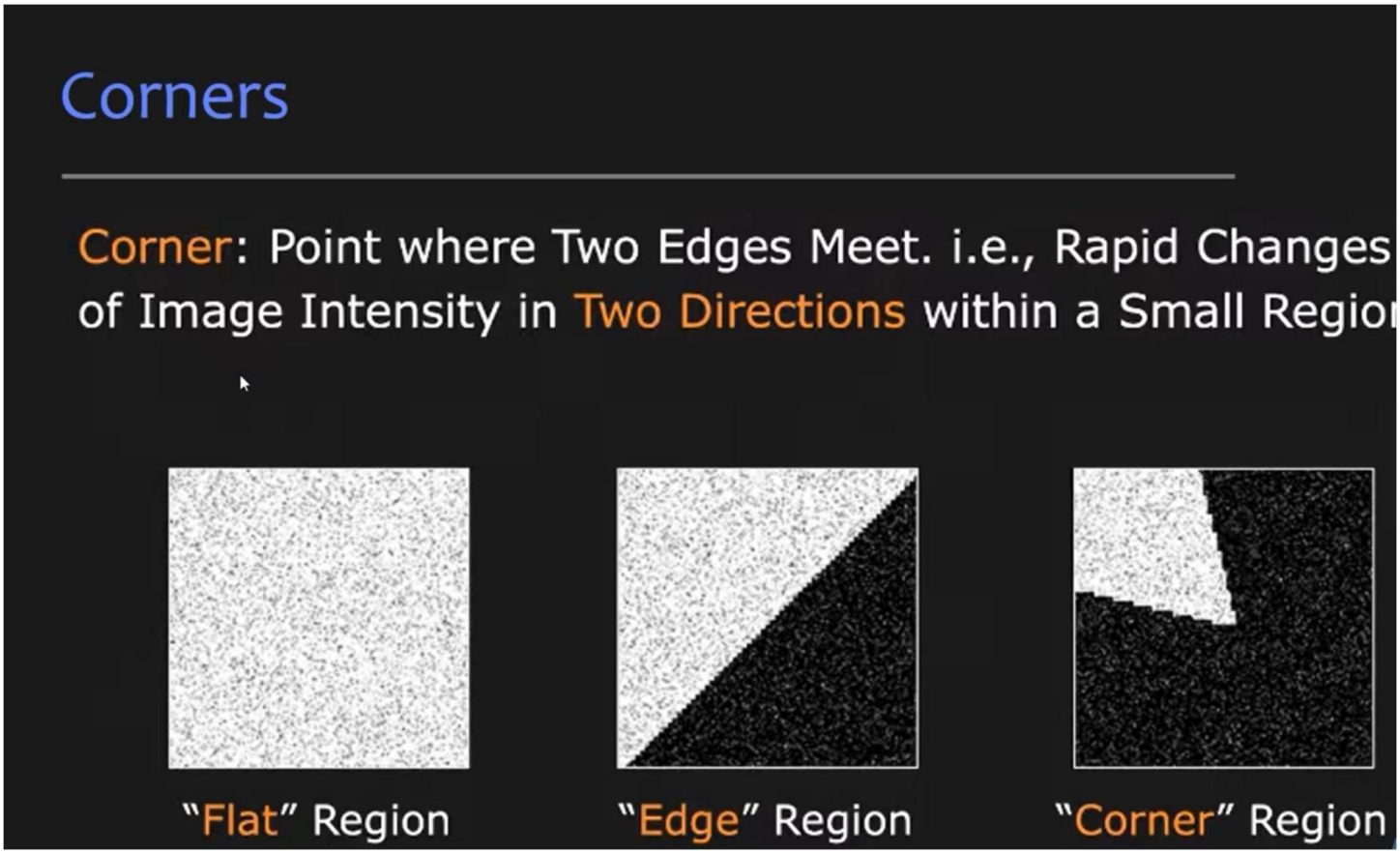
در نسخه‌های جدیدترOpenCV ، از الگوریتم‌های پیشرفته‌تر مانند **FAST**، **ORB**و **SIFT**نیز برای تشخیص گوشه‌ها و ویژگی‌ها استفاده می‌شود. این روش‌ها علاوه بر سرعت بالاتر، مقاومت بیشتری در برابر تغییرات مقیاس، چرخش و نور دارند. برای مثال، الگوریتم FAST (Features from Accelerated Segment Test) بسیار سریع است و در کاربردهایی مانند ردیابی در ویدئوهای بلادرنگ (Real-Time Tracking) استفاده می‌شود. به طور کلی، انتخاب روش مناسب به نوع مسئله، سرعت پردازش مورد نیاز و میزان دقت مدنظر بستگی دارد.

در کاربردهایی مثل ساخت تصاویر **پانوراما**، تشخیص گوشه‌ها نقش حیاتی دارد. برای اتصال چند تصویر و ایجاد یک نمای وسیع‌تر، باید نقاط مشترک (Keypoints) بین تصاویر مختلف پیدا شوند. گوشه‌ها به دلیل تغییرات شدید شدت روشنایی در دو جهت، بهترین گزینه برای این کار هستند، چون احتمال اشتباه کمتر است و نقاط متمایزی محسوب می‌شوند. الگوریتم‌هایی مانند **Harris**یا **Shi-Tomasi**در یافتن این نقاط کلیدی کمک می‌کنند و سپس با استفاده از روش‌هایی مثل **matching**و **homography estimation**، تصاویر به درستی روی هم قرار می‌گیرند. بنابراین، کیفیت نهایی یک پانوراما‌ی دقیق و بدون درز به شدت وابسته به شناسایی و تطبیق درست گوشه‌هاست.

الگوریتم **Harris Corner Detection**یکی از معروف‌ترین و قدیمی‌ترین روش‌ها برای شناسایی گوشه‌ها در تصاویر است. این الگوریتم در سال 1988 توسط Chris Harris و Mike Stephens معرفی شد و هنوز هم در بسیاری از کاربردهای پردازش تصویر استفاده می‌شود. گوشه‌ها نقاطی هستند که تغییرات شدیدی در شدت روشنایی تصویر در دو جهت (افقی و عمودی) دارند . Harris این موضوع را به صورت ریاضی مدل‌سازی کرد و با استفاده از یک تابع پاسخ (Corner Response Function) نقاط گوشه را مشخص کرد.

مبنای کار الگوریتم، **تحلیل تغییرات شدت روشنایی در اطراف یک نقطه** است. اگر یک پنجره کوچک روی تصویر در نظر بگیریم و آن را کمی در جهت‌های مختلف حرکت دهیم، تغییرات شدت روشنایی (intensity variation) نشان می‌دهد که آیا نقطه داخل آن یک گوشه، یک لبه یا ناحیه صاف است:

- اگر تغییرات در هر دو جهت کوچک باشد ← ناحیه یکنواخت.(flat)
- اگر تغییرات در یک جهت زیاد و در جهت دیگر کم باشد ← لبه.(edge)
- اگر تغییرات در هر دو جهت زیاد باشد ← گوشه.(corner)



برای محاسبه این تغییرات، Harrisاز ماتریس خود-همبستگی (Autocorrelation Matrix) یا همان ماتریس ساختار استفاده می‌کند که بر اساس مشتق‌های تصویر در دو جهت گرادیان‌های Ix و Iy ساخته می‌شود. این ماتریس، که به آن **Second Moment Matrix** هم می‌گویند، اطلاعاتی در مورد شدت تغییرات روشنایی در همسایگی یک پیکسل ذخیره می‌کند. سپس با محاسبه مقادیر ویژه (Eigenvalues) این ماتریس می‌توان نوع نقطه را تعیین کرد:

- دو مقدار ویژه کوچک ← ناحیه یکنواخت.
- یک مقدار ویژه بزرگ و دیگری کوچک ← لبه.
- دو مقدار ویژه بزرگ ← گوشه.

برای تشخیص عملی گوشه‌ها، الگوریتم Harris یک تابع پاسخ (R) تعریف می‌کند:

$$^2(trace(M)) \cdot k - det(M) = R$$

که در آن:

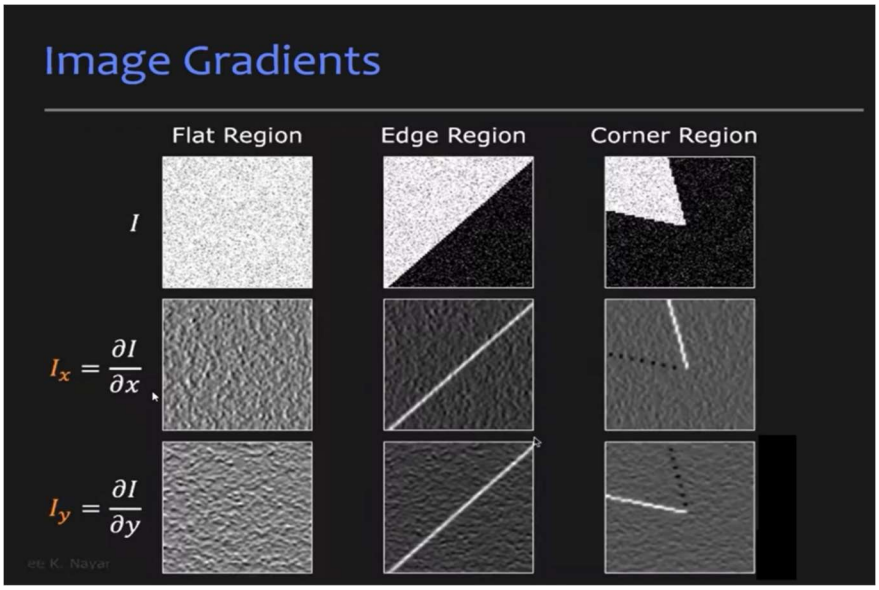
- Mماتریس ساختار است،
- det(M) حاصل‌ضرب مقادیر ویژه($\lambda 1 \times \lambda 2$) ،
- trace(M) مجموع مقادیر ویژه($\lambda 1 + \lambda 2$) ،
- kیک پارامتر تجربی است (معمولاً بین 0.04 تا 0.06).

اگر مقدار Rبزرگ و مثبت باشد ← گوشه،

اگر Rمنفی باشد ← لبه،

اگر Rنزدیک صفر باشد ← ناحیه یکنواخت.

مزیت اصلی روش Harris این است که نسبت به تغییرات چرخش تصویر مقاوم است، یعنی اگر تصویر بچرخد گوشه‌ها همچنان قابل شناسایی خواهند بود. با این حال، این الگوریتم به تغییر مقیاس (Zoom in/out) و تغییرات شدید نور حساس است. برای همین بعدها الگوریتم‌هایی مثل **Shi-Tomasi** و روش‌های مقاوم به مقیاس مانند **SIFT** و **SURF** معرفی شدند.



برای تشخیص گوشه ها می توان از کد زیر استفاده نمود:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('images/blox.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
gray = np.float32(gray)
dst = cv2.cornerHarris(gray,2,3,0.04)
#result is dilated for marking the corners, not important
dst = cv2.dilate(dst,None, iterations=1)
# Threshold for an optimal value, it may vary depending on the image.
img[dst>0.01*dst.max()]=[0,0,255]
plt.imshow(img[...,::-1]);
```

الگوریتم Harris :

```
cv2.cornerHarris(src, blockSize, ksize, k)
```

توضیحات پارامترها:

1. **src**
   - تصویر ورودی باید **Grayscale** و از نوع **float32** باشد.
   - به همین دلیل معمولاً از `cv2.cvtColor()` برای خاکستری کردن و از `np.float32` (برای تبدیل نوع داده استفاده می‌کنیم.
2. **blockSize**
   - اندازه پنجره همسایگی (neighborhood) برای محاسبه ماتریس M.
   - مثلاً وقتی مقدار 2 داده شود، الگوریتم روی بلوک‌های 2×2 پیکسل گرادیان را محاسبه می‌کند.
   - اگر کوچک انتخاب شود ⟶ حساسیت زیاد ولی پر از نویز.
   - اگر بزرگ انتخاب شود ⟶ نتیجه صاف‌تر اما ممکن است گوشه‌های ریز از دست بروند.
3. **ksize**
   - اندازه کرنل Sobel برای محاسبه گرادیان‌ها (مشتق‌ها).
   - معمولاً مقدار 3 مناسب است.
   - باید **عدد فرد** باشد 1 ، 3، 5، ....

4. **k**
   - ثابت حساسیت الگوریتم Harris.
   - مقدار پیش‌فرض بین **0.04** تا **0.06** توصیه می‌شود.
   - اگر کوچک انتخاب شود ⟶ الگوریتم نقاط بیشتری را گوشه فرض می‌کند.
   - اگر بزرگ انتخاب شود ⟶ فقط گوشه‌های قوی‌تر شناسایی می‌شوند.