دانشکدگان فنی دانشگاه تهران
دانشکده مهندسی نقشه برداری و اطلاعات مکانی

تمرین شناسایی نقاط گوشه

دانشجو:
علیرضا براهیمی

شماره دانشجویی:
810301017

استاد:
دکتر حسنلو

نیمسال اول سال تحصیلی 1401 – 1402

مقدمه :

تشخیص گوشه روشی است که در سیستم‌های بینایی کامپیوتر برای استخراج انواع خاصی از ویژگی‌ها و محتویات یک تصویر استفاده می‌شود. تشخیص گوشه اغلب در تشخیص حرکت، انطباق تصویر، ردیابی ویدئو، موزاییک تصویر، دوخت پانوراما، مدل‌سازی سه‌بعدی و تشخیص شی مورد استفاده قرار می‌گیرد. در این پروژه ما برای شناسایی نقاط گوشه از چهار الگوریتم harris ، sift ، moravec و shi-tosami استفاده کرده‌ایم که در ادامه درباره هر الگوریتم و نحوه عملکرد آن توضیح می‌دهیم .

Harris:

بر اساس این الگوریتم نقاط گوشه نقاطی هستند که اختلاف زیادی در شیب تغییر شدت تابش در تمامی جهات دارند . این الگوریتم که توسط آقای chris Harris و mike Stephens طراحی شده‌است با استفاده از رابطه زیر این قانون را پیاده‌سازی می‌کند .

$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \underbrace{[I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2$$

سپس با استفاده از رابطه‌ی زیر یک امتیاز برای هر پنجره در نظر گرفته می‌شود :

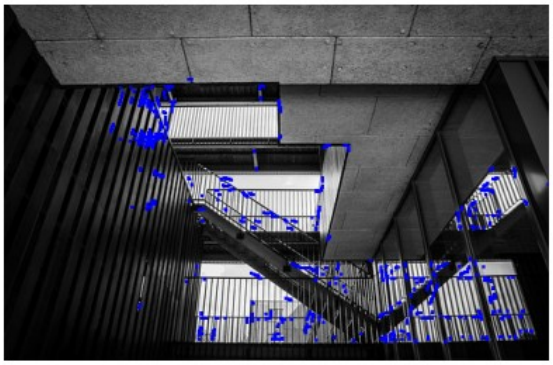
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

$$R = \det(M) - k(\text{trace}(M))^2$$

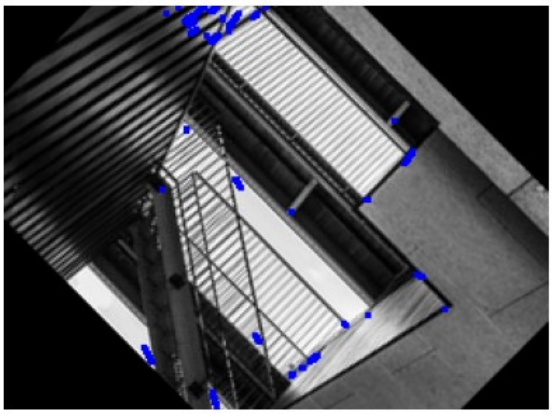
در مرحله‌ی بعد مقدار R به صورت زیر طبقه‌بندی می‌شود تا پنجره‌های صاف ، دارای لبه و دارای گوشه از یکدیگر متمایز شوند .

پنجره گوشه و یا لبه ندارد : اگر |R| خیلی کوچک باشد
پنجره دارای لبه است : اگر R<0
پنجره دارای گوشه است : اگر مقدار R خیلی بزرگ باشد

harris corner detection

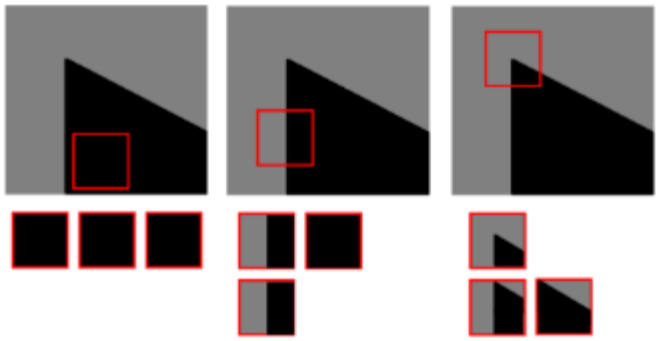


harris corner detection after transform



Moravec :

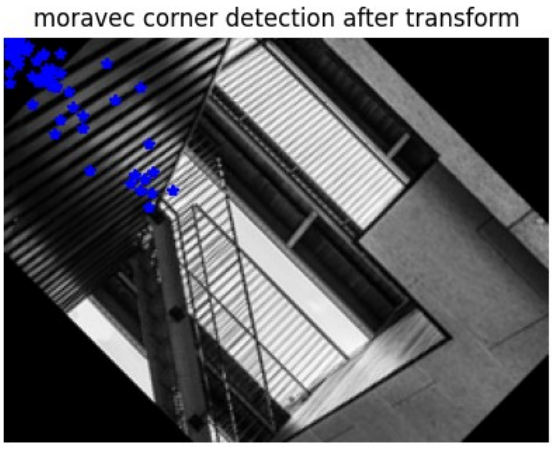
مبنای این الگوریتم این است که هنگامی که یک پنجره از عکس را در جهات مختلف یک پیکسل بررسی کنیم ، اگر پیکسل مورد نظر یک گوشه باشد تغییرات شدید خواهد بود .



در این روش قانون بالا بصورت زیر پیاده سازی می‌شود .

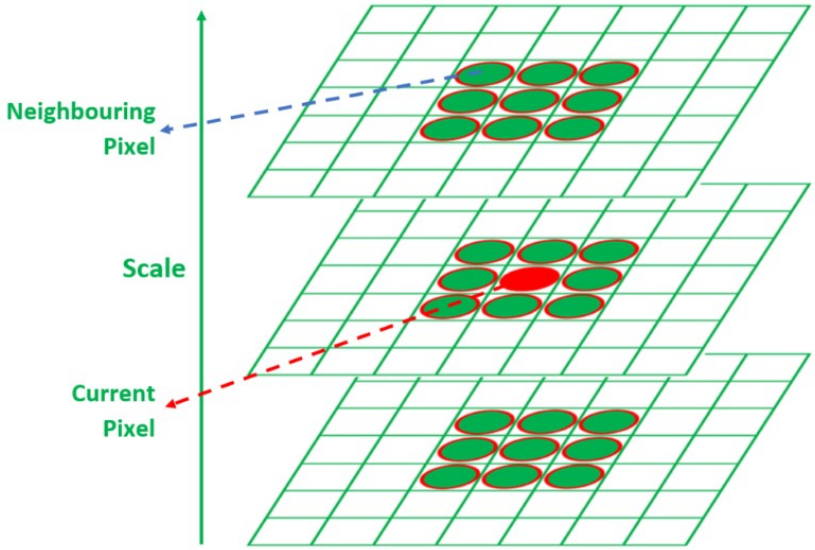
$$E_{m,n}(x,y) = \sum_{u,v} w_{m,n}(u,v)\left[f(u+x,v+y) - f(u,v)\right]^2$$

برای هر پیکسل مقدار min رابطه بالا نگه داشته می‌شود . در انتها بصورت محلی مقدار max F به عنوان نقطه گوشه در نظر گرفته می‌شود .



Sift :

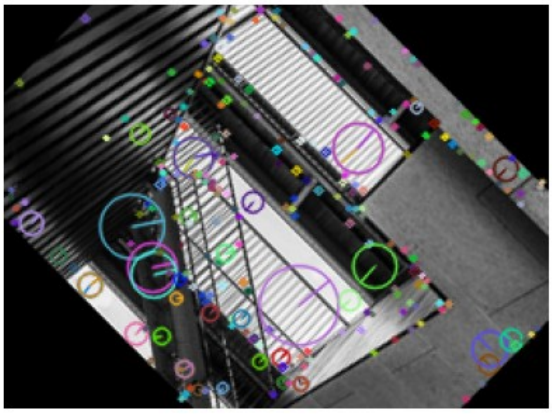
الگوریتم sift از الگوریتم های مورد استفاده در بسیاری از کاربردها مانند شناسایی اجسام و غیره است . مزیت این روش نسبت به روش harris این است که به ویژگه‌های لحظه عکسبرداری مانند depth ، view point و scale بستگی ندارد . بدین منظور این الگوریتم ابتدا عکس را به بعد scale-invariant انتقال می‌دهد . منطق استفاده از فضای scale در این روش این است که اگر یک پیکسل ، ویژگی مورد نظر ما مانند گوشه بودن را در scale های مختلف داشت آنگاه آن ویژگی را به پیکسل مورد نظر نسبت می‌دهیم . در این فضا ما می‌توانیم نقاط مورد نظر را با استفاده از گرفتن ماکسیمم یا مینیمم محلی از لاپلاسین گوسین به دست آوریم .



در مرحله بعد نقاط به دست آمده را پالایش می‌کنیم تا نقاط با سطح اطمینان بالاتر را به‌دست آوریم . برای اینکار لاپلاسین نقاط کلیدی را محاسبه می‌کنیم .
پس از پالایش نقاط کلیدی ، به هریک از نقاط باقی‌مانده یک جهت نسبت می‌دهیم . دلیل این کار این است که بتوانیم شناسایی این نقاط را نسبت به دوران بی تاثیر کنیم . برای اینکار در یک پنجره برای هر پیکسل با توجه به همسایگی‌های آن مقدار magnitude و direction را برای گرادیان همسایگی محاسبه می‌کنیم و با بدست آوردن histogram به هر پیکسل یک orientation نسبت می‌دهیم.
در مرحله آخر برای هر نقطه کلیدی یک توصیفگر با استفاده از همسایگی‌های آن تعیین می‌کنیم . از این توصیفگر می‌توان برای شناسایی نقطه استفاده کرد .

sift corner detection

sift corner detection after transform



Shi-tomasi :

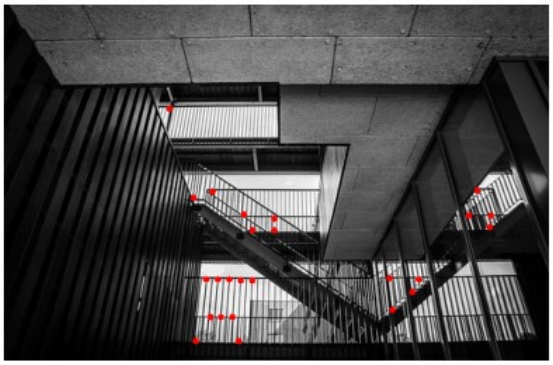
این روش همانند haris عمل می‌کند با این تفاوت که برای R از رابطه زیر استفاده می کند .

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

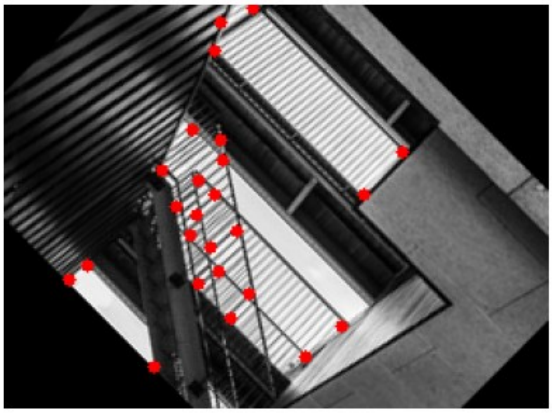
$$R = min(\lambda_1, \lambda_2)$$

استفاده از رابطه بالا باعث نگه‌داری نقاط قوی به عنوان نقاط گوشه می‌شود .

shi-tomasi corner detection



shi-tomasi corner detection after transform



```
import matplotlib.pyplot as plt
import cv2
from PIL import Image
def harris(image):
   img = image.copy()
   gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
   gray = np.float32(gray)
   dst = cv2.cornerHarris(gray, 2,21, 0.1)
   dst = cv2.dilate(dst, None)
   img[dst > 0.01 * dst.max()] = [0, 0, 255]
   return img


def moravec(img):
   thresh = 100
   gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
   rgb = cv2.cvtColor(gray.copy() , cv2.COLOR_GRAY2RGB)
   shifts = [(1 , 0) , (1 , 1) , (0 , 1) , (-1 , 1)]
   for y in range(1 , rgb.shape[1] - 1):
       for x in range(1 , rgb.shape[0] - 1):
           e = 100000
           for shift in shifts:
               diff = gray[x + shift[0] , y + shift[1]]
               diff = diff - gray[x , y]
               diff = diff * diff
               if diff < e :
                   e = diff
           if e > thresh :
               rgb[x , y] = (0 , 0 ,255)
   return(rgb)


def Sift(image):
   img = image.copy()
   gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
   sift = cv2.xfeatures2d.SIFT_create()
   kp = sift.detect(gray, None)
   img =
cv2.drawKeypoints(gray,kp,img,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEY
POINTS)
   return(img)


def shi_tomasi(image):
```

```
    img = image.copy()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    corners = cv2.goodFeaturesToTrack(gray,25,0.01,10)
    corners = np.int0(corners)
    return corners
```