بنا بر فرض سؤال، ما ۱۲۰ نقطه روی وتر مثلث و همین‌طور ۸۰ و ۶۰ نقطه روی ۲ ضلع دیگران داریم. در صورت سؤال اماده است که لبه های این مثلث را داریم، پس بنا بر این موضوع، 3 نقطه داریم که این 3 لبه را مشخص میکند.

فرض ما این است که این ۳ نقطه مشخص‌کننده لبه نیز جزو مجموعه نقاط واقع در ۱۲۰ نقطه وتر و نقاط باقی در اضلاع دیگر است.

همین‌طور ۱۰۰ نقطه نیز در خارج از اضلاع این مثلث داریم و بنابراین تعداد نقاط ما در کل صفحه برابر با ۳۶۰ است.

اگر ما بخواهیم حالتی را حساب کنیم که ضلع وتر را پیدا کنیم، در مرحله اول نیاز به پیداکردن نسبت تعداد نقاط inlier به تمام نقاط را داریم. تعداد نقاط وتر ما برابر با ۱۲۰ است و تعداد همه نقاط ما برابر با ۳۶۰ است، پس احتمال انتخاب یک نقطه از وتر برابر با ۰/۳۳ است. حال برای اینکه RANSAC نیاز به یک جفت نقطه دارد.

پس با فرض اینکه w نسبت تعداد نقاط inlier به تمام نقاط باشد و p احتمال یافتن یک مجموعه از نقاط بدون outlier باشد، آنگاه اگرk تعداد تکرار باشد، احتمال آنکه هیچ مجموعه درستی انتخاب نشده باشد برابر است با:

$$1 - p = (1 - w^2)^k$$

$$k = \frac{log(1-p)}{log(1-w^2)}$$

حال باتوجه‌به سؤال،p ما در مرحله اول ۰/۹ است و همین‌طور $w = \frac{1}{3}$ پس بنابر این مقدار تکرار حداقل برای اینکه به احتمال 90 درصد به جواب مناسب برسیم برابر با:

$$k = \frac{\log(1-0.9)}{\log(1-\frac{1}{9})} = \frac{-1}{-0.051} = 19.54$$

بنابراین، با ۲۰ بار امتحان به این احتمال می‌رسیم.

حال اگر p=0.99 باشد، مانند بالا می‌توانیم k را محاسبه کنیم:

$$k = \frac{\log(1-0.99)}{\log(1-\frac{1}{9})} = \frac{-2}{-0.051} = 39.09$$

بنابراین، با ۴۰ بار امتحان می‌توانیم به‌احتمال ۹۹ درصد برسیم.

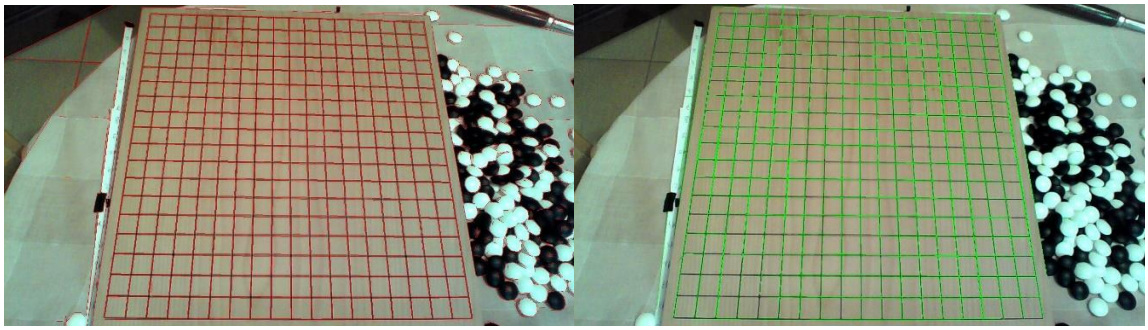
۲. پارامترهایی که در نتیجه ما تأثیر دارند، میزان thresholdهای ما هنگام استفاده از فیلتر canny است و همین‌طور minLineLength را نیز داریم که فقط خط‌هایی را به‌عنوان کاندیدای اصلی لحاظ می‌کنیم که بیشتر از این باشد و maxLineGap نیز نشان‌دهنده بیشترین مقدار مجاز برای Gapهای خط‌های کاندیدا است. مانند تابع بخش بالا، threshold ای داریم که در انباشتگر چند رأی لازم است تا این (rho، تتا) به‌عنوان یک خط در نظر گرفته شود. Rho نشان‌دهنده وضوح فاصله انباشتگر بر حسب پیکسل و تتا نشان‌دهنده وضوح زاویه انباشته بر حسب رادیان است.

در کد نوشته شده، مقدار این پارامترها برابر با:

```
cv2.HoughLinesP(rho=1,theta= np.pi/360,threshold=140,minLineLength=5, maxLineGap=15)
```

۳. کد نوشته شده LineSegmentDetector را استفاده کرده است. هدف از این الگوریتم یافتن نقاط ابتدا و انتهای پاره‌خط‌های موجود در تصویر است و هر پاره خط به‌جای ۲ پارامتر توسط ۴ پارامتر مشخص می‌شود و همین‌طور مزیت اصلی الگوریتم LSD آن است که به‌خوبی از جهت گرادیان استفاده می‌کند.

باتوجه‌به نتایج به‌دست‌آمده، این الگوریتم نتیجه بهتری نسبت به hough دارد :



عکس سمت راست حالتی است که ما از Probabilistic استفاده کردیم و دارای خط‌هایی است که از صفحه شطرنجی ما خارج شده‌اند؛ ولی در عکس سمت راست به دلیل اینکه پاره خط پیدا کردیم، نتیجه در دید اولیه بهتر است و هم طور تمامی خط‌های صفحه شطرنجی را برعکس عکس راست، پیدا کرده است.

در حالت کلی با LSD ما بهتر می‌توانیم به‌وسیله گرادیان، shape شکل را در بیاوریم.

منابع:

https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549

alyssaq/hough_transform: Hough Transform implementation in Python (github.com)

colors - How can I convert RGB to CMYK and vice versa in python? - Stack Overflow

RGB to YCbCr conversion (sistenix.com)