۱.

$p = \frac{\text{احتمال یافتن مجموعه بدون outlier}}{} = 0.9$

$w = \frac{\text{نقاط inlier}}{\text{تمام نقاط}} = \frac{\text{نقاط لبه وتر}}{\text{کل}} = \frac{۱۲۵}{۳۶۰} = \frac{۱}{۳}$

$K = \frac{\log(1-p)}{\log(1-w^2)} \simeq 19.55$

برای بدست آوردن ضلع وتر با احتمال 90٪ یا بیشتر لازم است حداقل 20 بار الگوریتم اجرا شود

$p = 0.99 \quad w = \frac{1}{3} \simeq 0.333 \Rightarrow K = \frac{\log(1-p)}{\log(1-w^2)} \simeq 39.1$

برای بدست آوردن ضلع وتر با احتمال 99٪ یا بالاتر لازم است حداقل 40 بار الگوریتم اجرا شود.

۲. تابع HoughLines

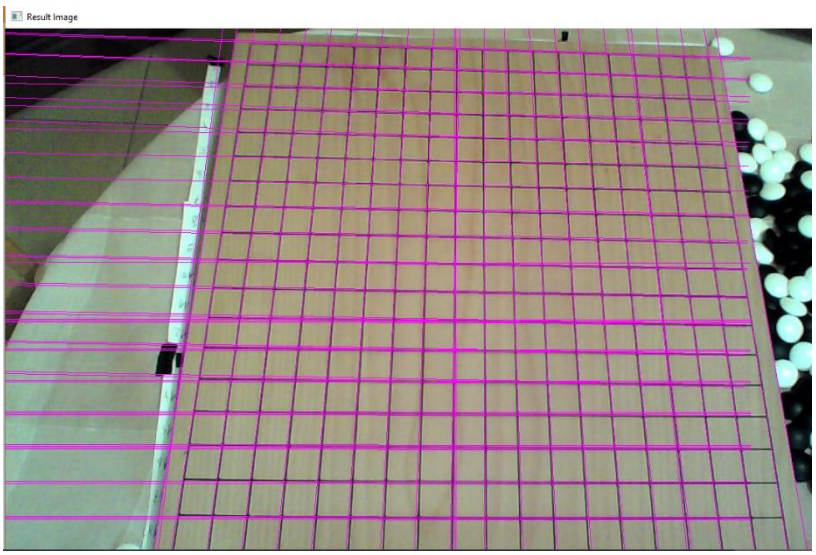
```python
# Read image
img = cv2.imread('LineDetection.jpg', cv2.IMREAD_COLOR)
# Convert the image to gray-scale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Find the edges in the image using canny detector
edges = cv2.Canny(gray, 50, 200)
# Detect points that form a line
lines = cv2.HoughLines(edges, 1, np.pi/180, 250)
# Draw lines on the image
for line in lines:
    r, theta = line[0]

    cos = math.cos(theta)
    sin = math.sin(theta)

    x1 = int(cos * r + 1000 * (-sin))
    y1 = int(sin * r + 1000 * cos)

    x2 = int(cos * r - 1000 * (-sin))
    y2 = int(sin * r - 1000 * cos)

    cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), 1)
# Show result
cv2.imshow("Result Image", img)
cv2.waitKey()
```



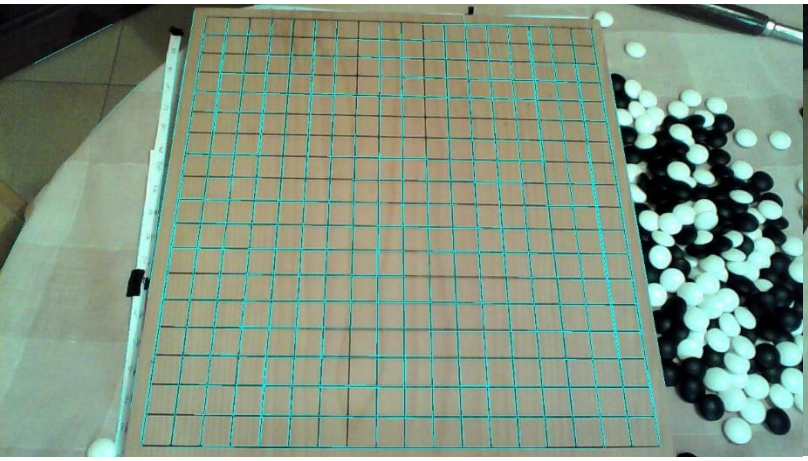
خروجی تابع HoughLines خطوط یافت‌شده است بطوری که هر خط بصورت آرایه‌ی تک‌عضوی [[r,theta]] خروجی داده می‌شود. پس از بدست‌آوردن دو نقطه دلخواه از خط، آن را رسم می‌کنیم. (اگر بجای ۱۰۰۰ ضریب کوچکتری می‌گذاشتیم خطوط طول کمتری می‌داشتند و عدد بزرگتری می‌گذاشتیم خطوط طولانی‌تر بودند چون دو نقطه‌ی دلخواه را با فاصله بیشتری انتخاب کرده بودیم)

تابع HoughLinesP

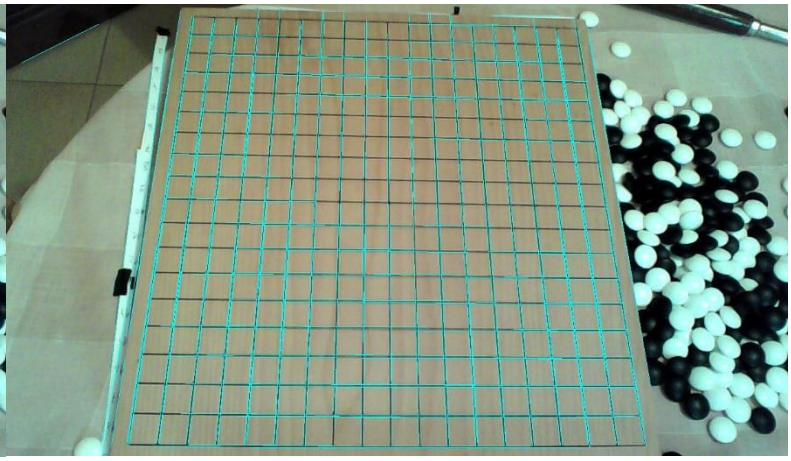
مقادیر مختلف زیادی برای پارامترها بررسی کردم. بهترین نتیجه را از این دو مقداردهی گرفتم:

```
lines = cv2.HoughLinesP(edges, 0.25, np.pi/360, 20, minLineLength=100, maxLineGap=15)
lines = cv2.HoughLinesP(edges, 0.25, np.pi/360, 20, minLineLength=80, maxLineGap=12)
```

بطور کلی با رزولوشن پیکسلی 0.25 و رزولوشن درجه $\frac{\pi}{360}$ نتایج بهتری می‌گرفتم.در هر دو مقداردهی حد آستانه رای‌گیری خط معتبر را ۲۰ رای گرفتم. کمینه طول موردقبول خطوط و حداکثر فاصله موردقبول نقطه از خط را در مقداردهی اول و دوم به ترتیب (۱۰۰، ۱۵) و (۸۰، ۱۲) گرفتم.



Result Image1.jpg



Result Image2.jpg

۳. تصویر دوبار بازشده یکبار بصورت سیاه‌وسفید تک‌کاناله و یکبار بصورت رنگی سه‌کاناله. روی تصویر تک‌کاناله خطوط پیدا شده‌اند و سپس نتیجه روی تصویر رنگی نمایش داده شده. طبق توضیحات داک OpenCV برای تشخیص خطوط در LineSegmentDetector از الگوریتمی که در این مقاله معرفی شده است استفاده می‌شود. این متود که به LSD مشهور است برخلاف کاری که در سوال قبلی کردیم نیازی به تنظیم کردن و پیداکردن پارامترهای مطلوب بصورت دستی ندارد و نتایج بسیار دقیق‌تری را در خروجی می‌دهد. از مقایسه خروجی این متود با خروجی که در قسمت قبلی از Hough گرفتیم می‌توانیم ببینیم خطوط مربوط به صفحه مشبک را بسیار دقیق‌تر پیدا کرده است البته علاوه بر خطوط صفحه بازی اصلی خطوط بیشتر دیگری را در حاشیه‌های تصویر (مثلا روی مهره‌های بازی) پیدا کرده است که شاید هدف ما از اجرای الگوریتم نبوده اما به هر صورت خطوط تصویر را بسیار دقیق و بهتر از ترکیب Canny و Hough که در قسمت قبل استفاده کردیم تشخیص داده است.

منابع:

سوال ۱.الف: Hough Transform using OpenCV | LearnOpenCV

Line detection in python with OpenCV | Houghline method - GeeksforGeeks

سوال ۴: colors - How can I convert RGB to CMYK and vice versa in python? - Stack Overflow