تمرین ۶

سینا اسکندری ۹۷۵۲۱۰۵۴

۱-

w = 120 / (120+80+60+100) = 1/3

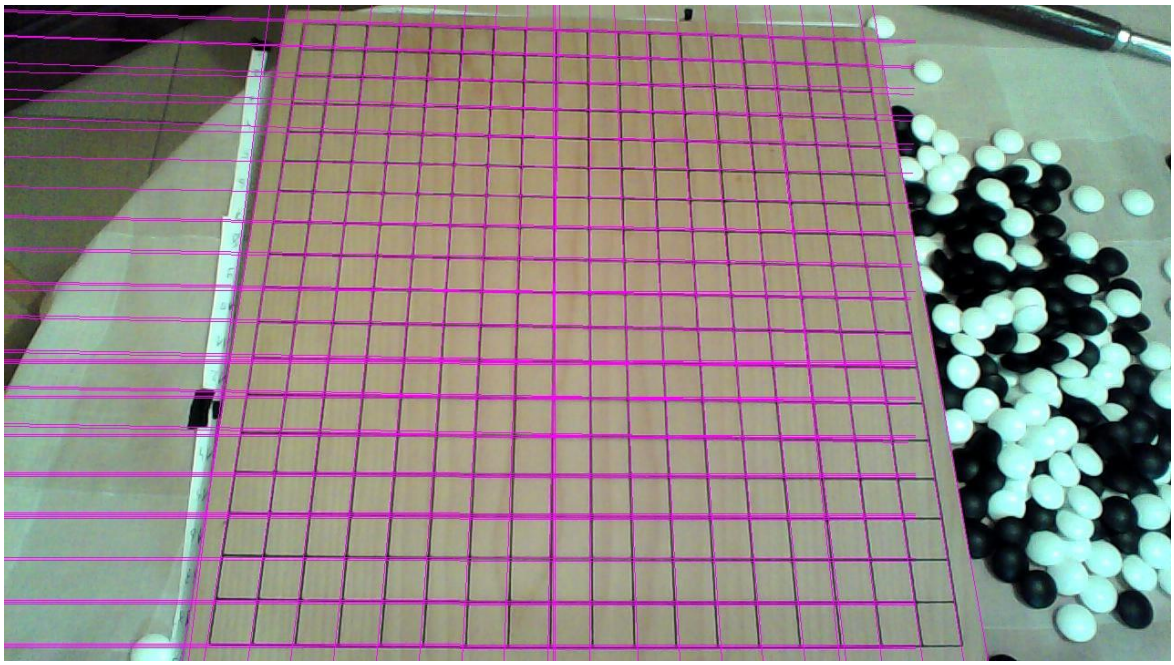
p = 0.9: $k = \frac{\log(1-p)}{\log(1-w^2)} = \frac{\log(0.1)}{\log(8/9)} \approx 20$

p = 0.99: $k = \frac{\log(1-p)}{\log(1-w^2)} = \frac{\log(0.01)}{\log(8/9)} \approx 39$

۲-

```python
#TODO
img = cv2.imread('LineDetection.jpg', cv2.IMREAD_COLOR)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 50, 200)
lines = cv2.HoughLines(edges, 1, np.pi/180, 250)
dst = img.copy()
for i in range(0, len(lines)):
    rho = lines[i][0][0]
    theta = lines[i][0][1]
    a = np.cos(theta)
    b = np.sin(theta)
    x0 = a * rho
    y0 = b * rho
    pt1 = (int(x0 + 1000*(-b)), int(y0 + 1000*(a)))
    pt2 = (int(x0 - 1000*(-b)), int(y0 - 1000*(a)))
    cv2.line(dst, pt1, pt2, (255,0,255))
cv2.imwrite('2a.jpg', dst)
plt.imshow(cv2.cvtColor(dst, cv2.COLOR_BGR2RGB))
plt.xticks([])
plt.yticks([])
```

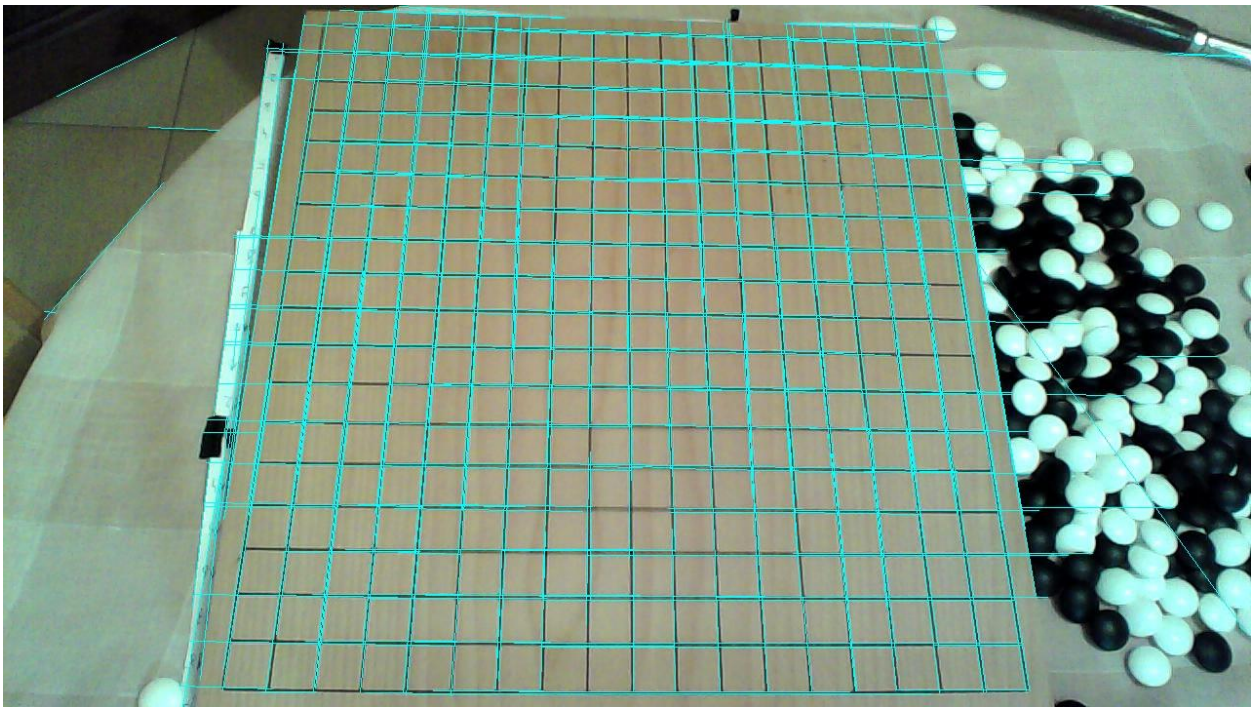
ابتدا با استفاده از canny لبه های تصویر را بدست می آوریم و با استفاده از HoughLines خطوط را محاسبه میکنیم.



برای استفاده از الگوریتم probabilistic hough transform به اینگونه عمل می کنیم:

```
#TODO
dst2 = img.copy()
lines = cv2.HoughLinesP(edges, 1, np.pi/180, 100, minLineLength=10, maxLineGap=30)
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv2.line(dst2, (x1, y1), (x2, y2), (255, 255, 0))
cv2.imwrite('2b.jpg', dst2)
plt.imshow(cv2.cvtColor(dst2, cv2.COLOR_BGR2RGB))
plt.xticks([])
plt.yticks([])
```

که minLineLength حداقل اندازه خطوط و maxLineGap بیشترین فاصله نقاط روی یک خط را مشخص می کنند.



۳-

این کد از الگوریتم LSD استفاده می کند که نقاط ابتدا و انتها پاره خط های موجود خطوط را پیدا می کند. این الگوریتم به خوبی از جهت گرادیان استفاده میکند و نتیجه بدست آمده در این حالت از حالت قبل بهتر می باشد.

۴-

با استفاده از فرمول های موجود و تبدیل مقیاس های RGB و CMYK توابع زیر را تعریف می کنیم.

```
def rgb_to_cmyk(r, g, b, RGB_SCALE = 255, CMYK_SCALE = 100):

  #TODO
  r /= RGB_SCALE
  g /= RGB_SCALE
  b /= RGB_SCALE

  k = 1 - max(r, g, b)

  c = round(((1 - r - k) / (1 - k)) * CMYK_SCALE)
  m = round(((1 - g - k) / (1 - k)) * CMYK_SCALE)
  y = round(((1 - b - k) / (1 - k)) * CMYK_SCALE)
  k = round(k * CMYK_SCALE)
  return c, m, y, k
```

```python
def cmyk_to_rgb(c, m, y, k, CMYK_SCALE = 100, RGB_SCALE = 255):

  #TODO
  c /= CMYK_SCALE
  m /= CMYK_SCALE
  y /= CMYK_SCALE
  k /= CMYK_SCALE

  r = round(((1 - k) - c * (1 - k)) * RGB_SCALE)
  g = round(((1 - k) - m * (1 - k)) * RGB_SCALE)
  b = round(((1 - k) - y * (1 - k)) * RGB_SCALE)

  return r, g, b
```

۵-

با استفاده از فرمول های موجود در صفحه ۲۳ و ۲۴ و ۲۹ اسلاید جلسه ۱۰ به صورت زیر مقادیر را بدست می آوریم.

```python
#TODO
(r, g, b) = (150, 65, 200)
theta = (np.arccos((r - g + r - b) / (2 * np.sqrt((r - g) ** 2 + (r - b) * (g - b))))) * (180 / np.pi)
H = theta if b <= g else 360 - theta
S = 1 - 3 * (min(r, g, b) / (r + g + b))
I = (r + g + b) / 3
V = max(r, g, b)
L = (max(r, g, b) + min(r, g, b)) / 2
Y = 0.299 * r + 0.587 * g + 0.114 * b
print(H, S, I, V, L, Y)
```

منابع:

https://learnopencv.com/hough-transform-with-opencv-c-python/

https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html

https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga46b4e588934f6c8dfd509cc6e0e4545a

https://docs.opencv.org/4.x/db/d73/classcv_1_1LineSegmentDetector.html

https://www.rapidtables.com/convert/color/rgb-to-cmyk.html