Computer vision

# Homework 6

Parsa Eissazadeh - 97412364

### سوال 1

احتمال آن که هیچ مجموعه درستی انتخاب نشده است طبق سوال برابر است با 10 درصد . طبق فرمول داریم :

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

1- p = 10% = 0.1

حال نیاز داریم به محاسبه $w^2$ . مقدار w برابر است با نسبت inlier ها به کل است . از آنجایی که 120 لبه برای وتر پیدا شده و تعداد کل لبه های پیدا شده برابر است با 100 + 60 + 80 + 120 داریم :

W = 120 / 360 = ⅓

به کمک google colab مقدار را حساب میکنم :

```
# Q1 calculations
p = .9
w = 1/3
k = np.log(.1) / np.log(1-np.power(w,2))
k
```

```
19.549378059091072
```

در نتیجه حداقل 20 بار . برای دقت 99 درصدی ، 1-p مقداری برابر با 0.01 میگیرد که همانند بالا تعداد تکرار های لازم را حساب میکنیم :

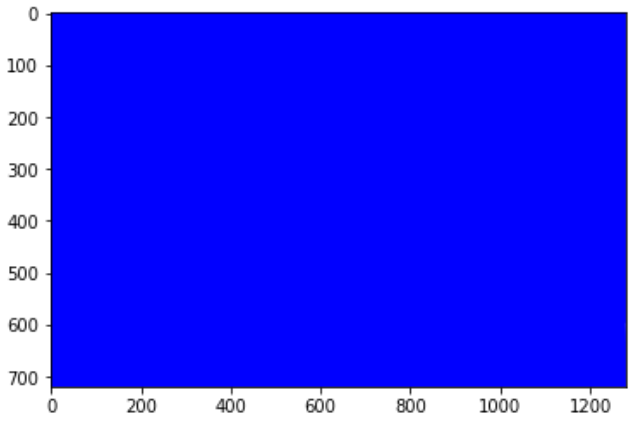
```
# Q1 calculations
p = .99
w = 1/3
k = np.log(1-p) / np.log(1-np.power(w,2))
k
```

```
39.098756118182145
```

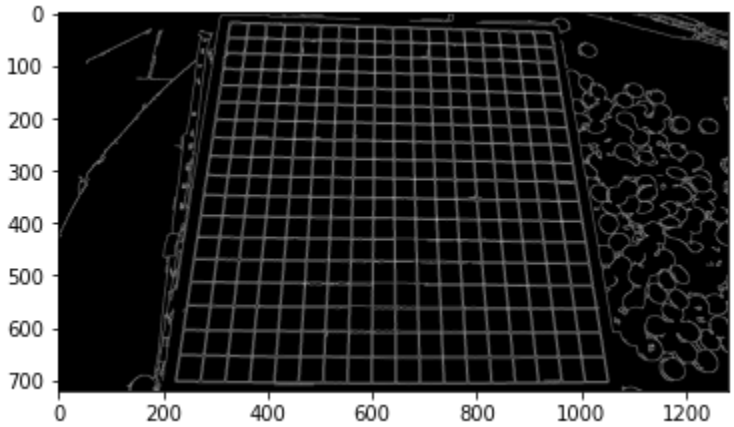
تعداد 40 دفعه حداقل تعداد لازم برای تکرار است .

## سوال 2

در ابتدا روی کل عکس hough را اعمال کردم و به نتیجه زیر رسیدم :




از آنجایی که تصویر ناواضح است ابتدا به کمک لبه یاب Canny ، لبه ها را استخراج می کنیم سپس روی آنها متد را اعمال می کنیم . آستانه های Canny را 50 و 200 میگذاریم .



نتیجه قابل قبول است . حال میرویم سراغ تشخیص خطوط . متد hough سه پارامتر می پذیرد ، ، rho ، tetha treshold و عکس ورودی .

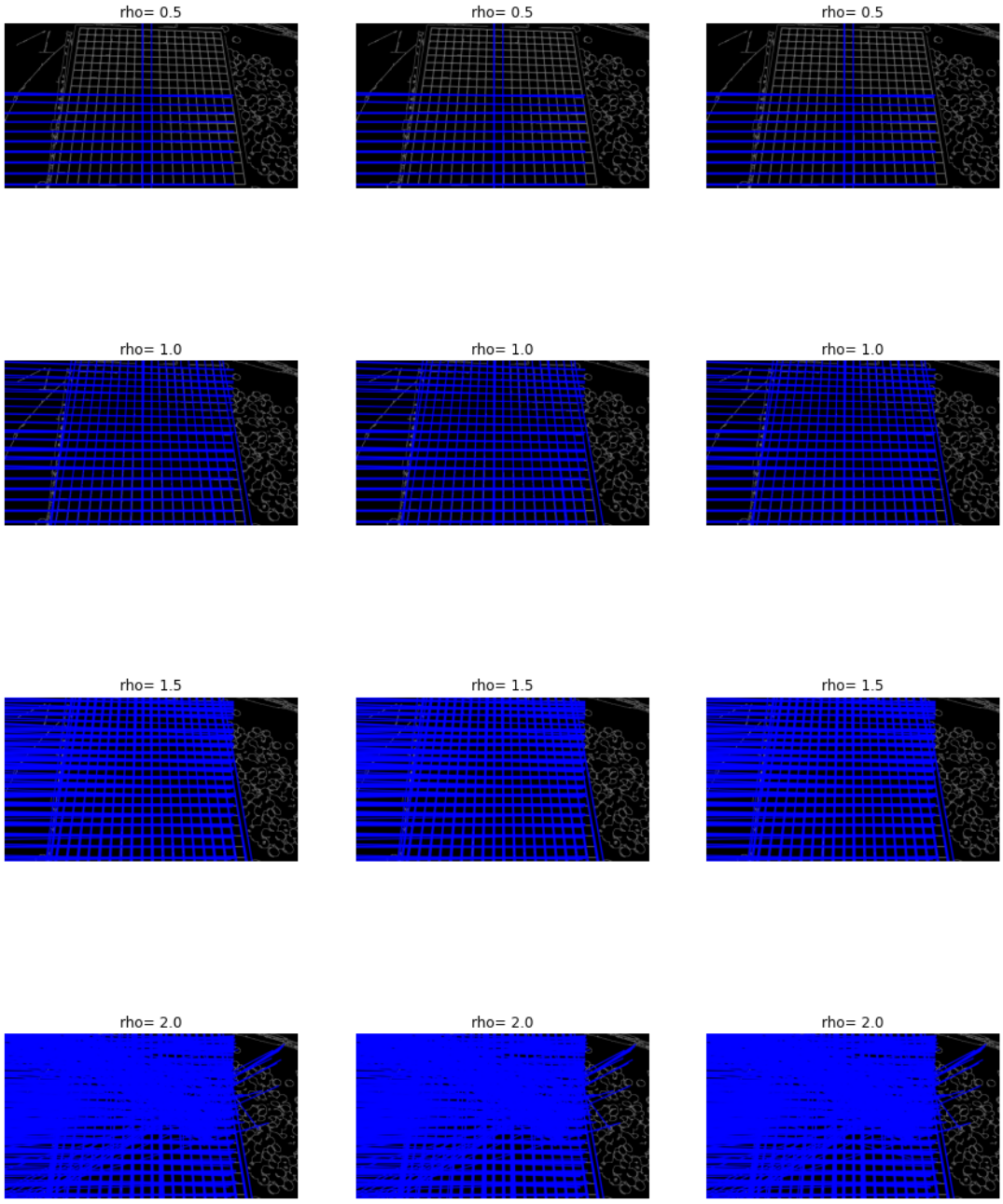
به کمک treshold کمترین تعداد رای را تنظیم میکنیم که طبق گفته سوال 250 است . rho و tetha دامنه ای را تعیین می کند که در آن به دنبال تایید کننده یک خط میگردیم . برای آن ها مقادیر مختلف ست کردم و خط ها را رسم کردم . نتیجه ( شکل در صفحه ی بعد ) .

مشاهده شد که tetha تاثیر زیادی ندارد و rho بسیار مهم است و همچنین واضح ترین نتیجه را rho با مقدار 1 داشت و tetha با مقدار 1 درجه رادیان داشت .

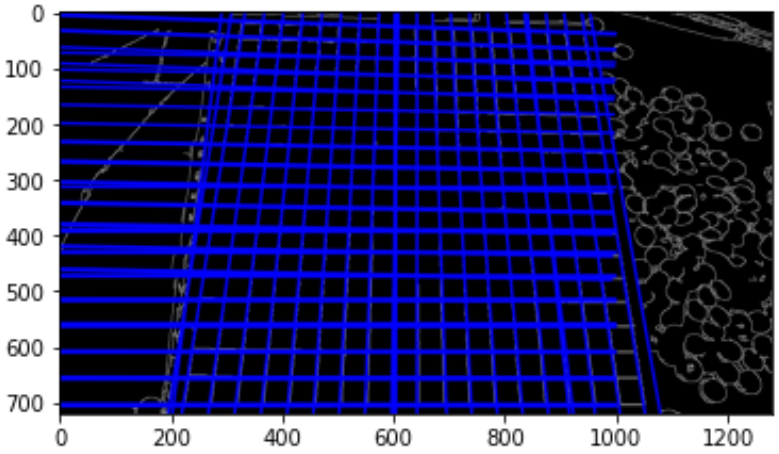
برای رسم خطوط نیاز است تمام تبدیلاتی که hough انجام می دهد را معکوس انجام دهیم . نیاز داریم تا نقطه ها را از فضای rho,tetha به فضای x,y ببریم . برای همین داریم :

$$X = \cos(tetha) * rho$$

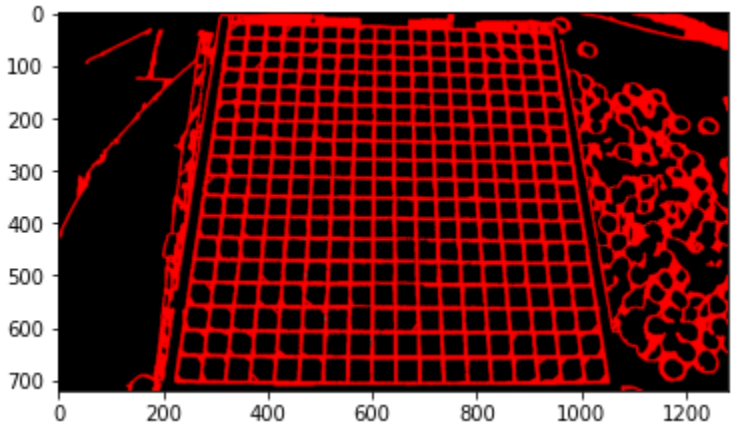
$$Y = \sin(tetha) * rho$$




بهترین را انتخاب میکنیم :




در تبدیل احتمال hough ، با کم و زیاد کردن پارامتر ها به خروجی زیر رسیدیم :




برای پیاده سازی تبدیل هاف در ابتدا باید هر x و y را به یک tetha و یک rho ای تبدیل کنیم . همچنین به یک accumulator ای احتیاج داریم که تعداد رای های هر خط را بشمارد . هر خط یک  دوتایی tetha و rho است در نتیجه هر خانه در accumulator به یک خط اختصاص دارد .

برای هر خط فرمول احتساب rho و tetha معلوم و واضح است . ولی نیاز داریم ابعاد accumulator را اضافه کنیم . برای تعیین این ابعاد از لینک زیر کمک گرفتم :

https://github.com/alyssaq/hough_transform/blob/master/hough_transform.py

کد :

```
# Hough implementation
edges = cv2.Canny(im, 50, 200)
cdst = cv2.cvtColor(edges, cv2.COLOR_GRAY2BGR)

width,height = edges.shape

# Defining accumulator
diag_len = int(round(math.sqrt(width * width + height * height)))
rhos = np.linspace(-diag_len, diag_len, diag_len * 2)

accumulator = np.zeros((2 * diag_len, 180), dtype=np.uint8)

for i in range(width):
  for j in range(height):
    if edges[i,j] < 128 :
      continue

    for tetha in range(0,180):
      rho = i * np.cos(tetha) + j * np.sin(tetha)
      accumulator[rho , tetha] += 1
```

## سوال 3

## سوال 4

مدل رنگ CMYK به شکل زیر حساب می شود :

$$K = 1 - max(R, G, B)$$

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1-K \\ 1-K \\ 1-K \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

در رابطه با k داریم :

$$1 - K = 1 - ( 1 - max(R,G,B) ) = max(R,G,B)$$

پس ما در هر نقطه باید max(R,G,B) را پیدا کنیم و اختلافش با R,G و B را حساب 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 - np.max((r,g,b))

  colors = np.full(3 , 1-k) - [r,g,b]
  colors *= CMYK_SCALE

  c,m,y = colors.astype(int)

  return c, m, y, (k * CMYK_SCALE).astype(int)
```

اما بعد از اجرای این کد به نتیجه مطلوب نرسیدیم . طبق لینک زیر باید در نهایت همه رنگ ها را تقسیم بر 1- k بکنیم :

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

کد نهایی :

```
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 - np.max((r,g,b))

  colors = np.full(3 , 1-k) - [r,g,b]
  colors *= CMYK_SCALE
  colors /= (1-k)

  c,m,y = colors.astype(int)

  return c, m, y, (k * CMYK_SCALE).astype(int)
```

برای تبدیل cmyk به rgb معکوس همه کار های قبلی را انجام میدهیم . کد :

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

  #TODO
  c,m,y,k = np.array([c,m,y,k])/ CMYK_SCALE

  colors = np.array([c,m,y])
  colors = colors * (1-k)
  colors = 1 - k - colors
  colors *= RGB_SCALE
  colors = colors.astype(int)

  r,g,b = colors

  return r, g, b
```

## سوال 5

سوال نکته پیچیده ای نداشت صرفا فرمول ها را به کمک python و numpy پیاده سازی کردیم :

```python
def compute_h(r,g,b):
  h_surat = r-g + r - b
  h_makhraj = np.power((r-g),2) + (r-b)*(g-b)
  h_makhraj = np.sqrt(h_makhraj)
  h_makhraj *= 2

  h = h_surat / h_makhraj
  h = np.arccos(h)

  return h
```

```python
#TODO
colors = np.array([150,65,200])
H = compute_h(150,65,200)
S = 1 - 3 * np.min(colors) / np.sum(colors)
I = np.average(colors)
V = np.max(colors)
L = (np.max(colors) + np.min(colors))/2
Y = 0.299*colors[0] + 0.587*colors[1] + 0.114*colors[2]

print("H: ", H)
print("S: ", S)
print("I: ", I)
print("V: " , V)
print("L: " , L)
print("Y: " , Y)
```

```
H:  1.422216023754124
S:  0.5301204819277108
I:  138.33333333333334
V:  200
L:  132.5
Y:  105.80499999999999
```

با تشکر !!