



دانشگاه صنعتی امیر کبیر

(پلی تکنیک تهران)

نام و نام خانوادگی:

پیمان هاشمی

شماره دانشجویی:

400131032

شماره تمرین:

تمرین شماره 5

درس:

تصویر پردازش رقمی

سوال (1)

پیمان هاشمی ۴۰۱۳۱۰۳۲

w B w B w B

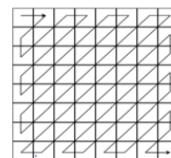
۲ → ۴ → ۲ → "EOL"
 ۱ → ۱ → ۴ → ۱ → ۱ → "EOL"
 ۱ → ۱ → ۱ → ۱ → ۲ → ۱ → "EOL"
 ۱ → ۱ → ۱ → ۱ → ۲ → ۱ → "EOL"
 ۴ → ۱ → ۱ → ۱ → "EOL"
 ۱ → ۲ → ۲ → ۱ → "EOL"
 ۱ → ۱ → ۴ → ۱ → ۱ → "EOL"
 ۲ → ۴ → ۲ → "EOL"

(a)

zig-zag → تداوم، تکرار

(b)

۳ → ۴ → ۲ → ۲ → ۱ → ۱ → ۱ → ۲ → ۲ → ۱ → ۱
 → ۱ → ۴ → ۱ → ۳ → ۱ → ۱ → ۱ → ۲ → ۳ → ۱
 → ۴ → ۱ → ۱ → ۲ → ۱ → ۲ → ۳ → ۲ → ۲ → ۴ → ۳ → "EOL"



(c) در این سوال اشاره نشده که Huffman code برای چیست a است یا b
 ((ب.ترجمه به سوال برای چیست a در نظریهٔ سرچ که تنها w و B داریم، در نظریه))

$$w = 20 \quad B = 16 \quad = \quad w = \frac{20}{24} \quad B = \frac{16}{24}$$

(d) این چیست هم با ترجمه که run length همین سوال برای چیست a

در نظریهٔ سرچ (نظریه)

۱	۲۳	$\frac{23}{44}$	۱	۲۳	
۲	۷	$\frac{7}{44}$	EOL	۸	
۳	۱	$\frac{1}{44}$	۲	۷	
۴	۵	$\frac{5}{44}$	۴	۵	
"EOL"	۸	$\frac{8}{44}$	۳	۱	

	Huffman Coding
۱	۰
EOL	۱۱
۲	۱۰۰
۴	۱۱۱۰
۳	۱۱۱۱

(e)

$$\text{entropi} \sum_i^n P(m_i) \log \frac{1}{P(m_i)}$$

$$H = - \left(\frac{22}{44} \cdot \log \frac{22}{44} + \frac{1}{44} \cdot \log \frac{1}{44} + \frac{1}{44} \cdot \log \frac{1}{44} + \frac{2}{44} \cdot \log \frac{2}{44} + \frac{1}{44} \cdot \log \frac{1}{44} \right)$$

$$H = 0.544$$

$$\text{Average number of bits} = \frac{22}{44} \times 1 + \frac{1}{44} \times 2 + \frac{1}{44} \times 2 + \frac{2}{44} \times 2 + \frac{1}{44} \times 2 = 1.9$$

(g)

0 → 0

22 → 44(0) + 22

20 → 44(0) + 20

14 → 44(0) + 14

48 → 44(0) + 48

44 → 44(1) + 0

10 → 44(1) + 10

140 → 44(2) + 12

220 → 44(2) + 28

180 → 44(2) + 02

200 → 44(2) + 18

120 → 44(2) + 06

140 → 44(2) + 22

210 → 44(2) + 18

200 → 44(3) + 43

0	22	14	48
20	14	180	14
44	220	180	200
140	140	210	200

محل می‌خواهیم تنها با ۲ بیت که کنیم باید اعداد باقی‌مانده در جمع باقی‌مانده را به بالا

۱۰ باینری که در کنیم (در هر بالای ۳۲ باینری به سمت بالا و اگر ۳۲ باینری به سمت پایین) را که نه‌تاری کنیم.

0 → 0

22 → 1

20 → 0

14 → 0

48 → 1

44 → 1

10 → 1

140 → 2

220 → 3

180 → 3

200 → 3

120 → 2

140 → 3

210 → 3

200 → 4

0	1	0	1
0	2	1	0
1	3	3	3
2	3	3	4

(i)

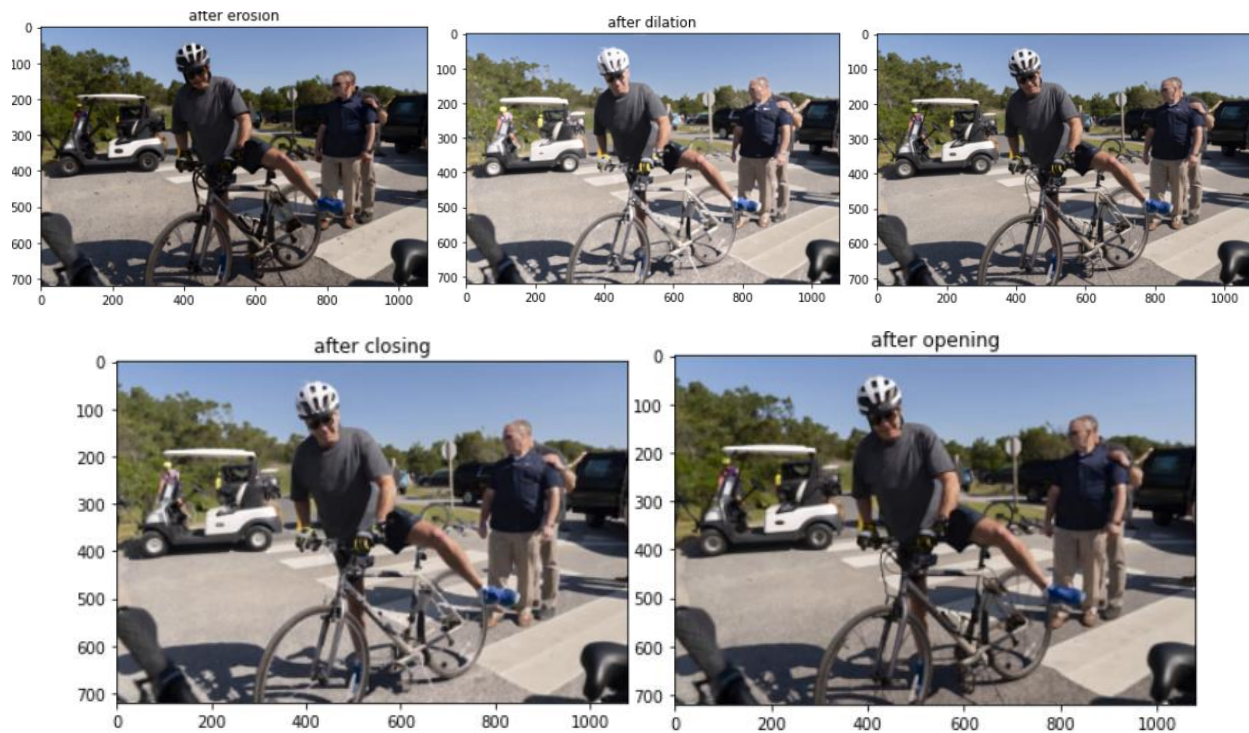
$$11at - 20t - r212 - 257914$$

(j)

18	18	4	1	3	8	1	3	2	14	18	14	3	7	10	14	18	14
----	----	---	---	---	---	---	---	---	----	----	----	---	---	----	----	----	----

سوال (2)

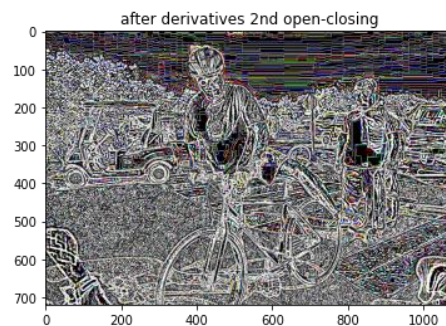
تصویر 1:



(A)

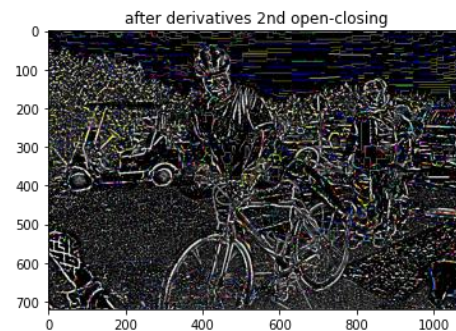
برای derivatives دوم با استفاده از erosion و dilation برابر زیر است:

```
result_g = (result_dilate/2) + (result_erode/2)
result_g = np.round(result_g, decimals = 0, out = None)
result_g = np.array(result_g, dtype=np.uint8)
result_a = img1 - result_g
```



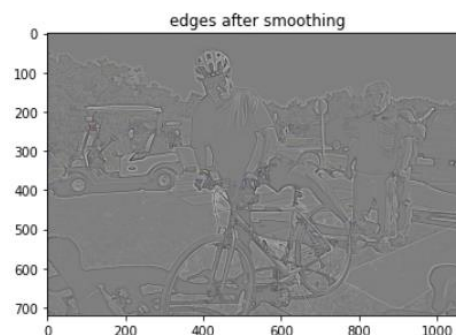
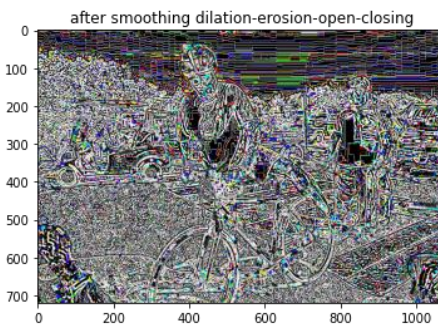
(B)

```
open_a = cv2.morphologyEx(img1, cv2.MORPH_OPEN, kernel)
open_close_a = cv2.morphologyEx(open_a , cv2.MORPH_CLOSE, kernel)
result_b = img1-open_close_a
```



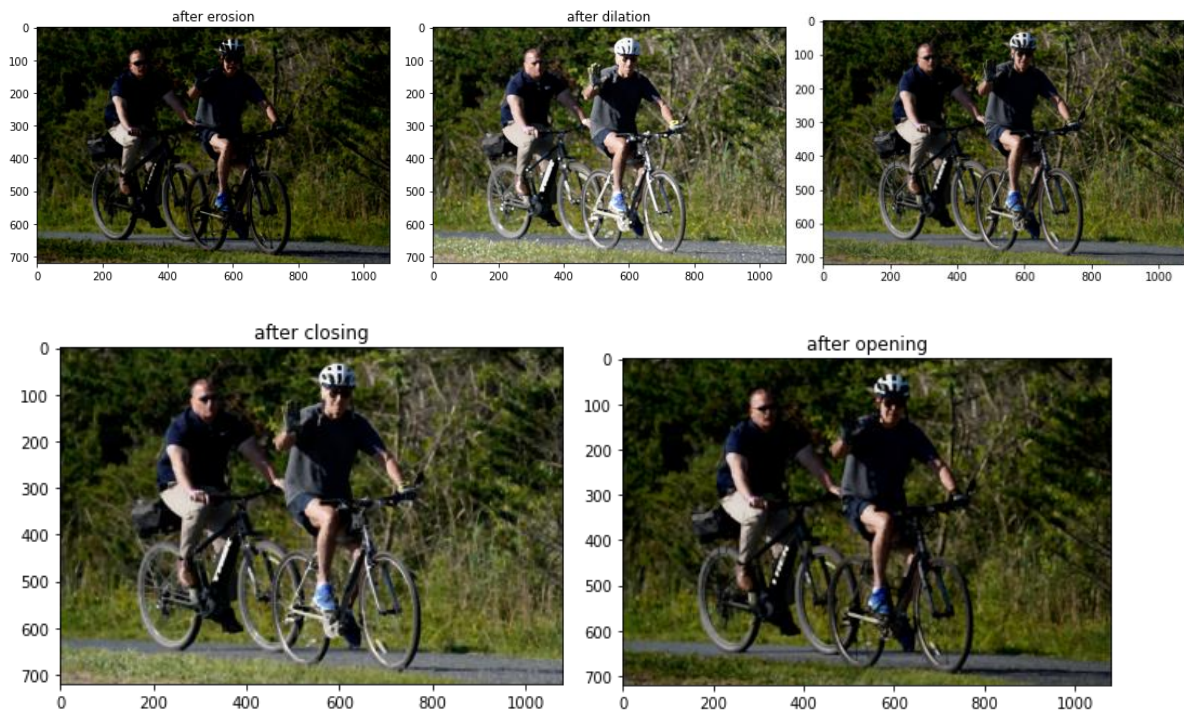
(C)

```
open_a = cv2.morphologyEx(result_a, cv2.MORPH_OPEN, kernel)
open_close_b = cv2.morphologyEx(open_a , cv2.MORPH_CLOSE, kernel)
result_c = result_a-open_close_b
```



در این قسمت با توجه به نتایج بدست آمده، لبه های مشخص شده ایی که از opening و closing بدست آمده است وضوح بیشتری دارند و کمتر نویز دارد و سپس نتایج بدست آمده از قسمت a نیز دارای مشخصات واضح تری نسبت به قسمت C میباشد.

تصویر دوم:



(D)

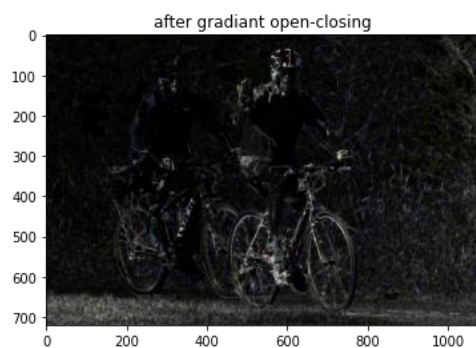
گرادیان با استفاده از erosion و dilation

```
result_d= result_dilate2- result_erode2
```



(E)

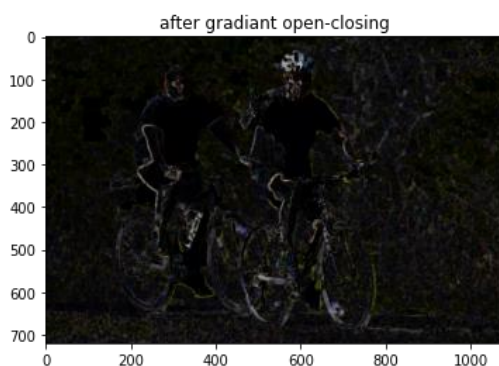
```
result_e= result_closing2- result_open2
```



(F)

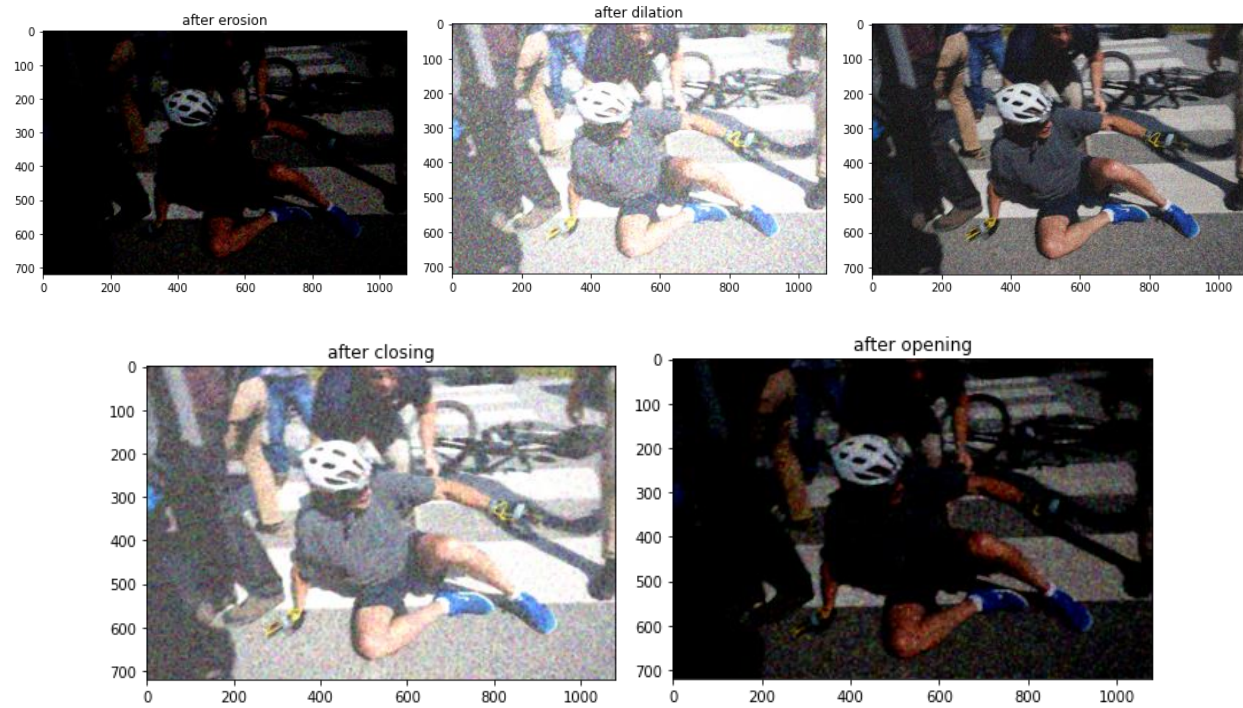
```
result_f= result_closing22- result_open22
```

از نتیجه قسمت d، opening و closing میگیریم و داریم:



در این قسمت نتایج بدست آمده از قسمت d بسیار مناسب است چرا که لبه ها را به خوبی نشان میدهد. سپس قسمت e نیز بهتر از قسمت f میباشد چرا که لبه ها موجود در دوچرخه واضح تر دیده میشود.

تصویر سوم:



(G)

```
result_g = (result_dilate3/2) + (result_erode3/2)
```



(H)

```
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5,5))
result_h = cv2.morphologyEx(img3, cv2.MORPH_OPEN, kernel)
result_h = cv2.morphologyEx(result_h, cv2.MORPH_CLOSE, kernel)
```

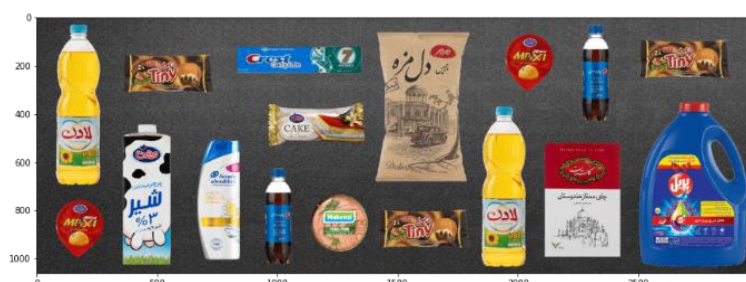



در این قسمت به نظر میرسد نتایج بدست آمده از قسمت h بهتر از قسمت g باشد ولی باید به این نکته توجه داشت که نتیجه قسمت h از قسمت g تاریک تر میباشد.

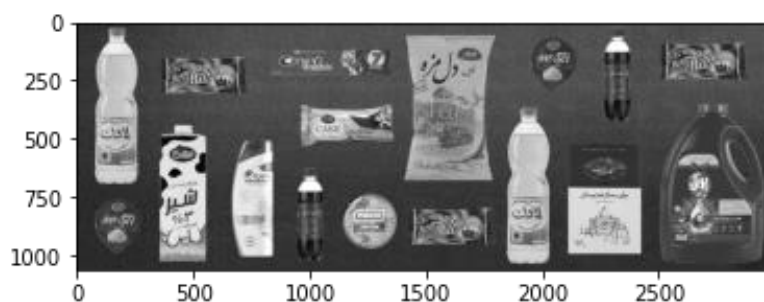
سوال (3)

(A)

برای جواب سوال 1 در ابتدا عکس مربوطه را لود میکنیم.

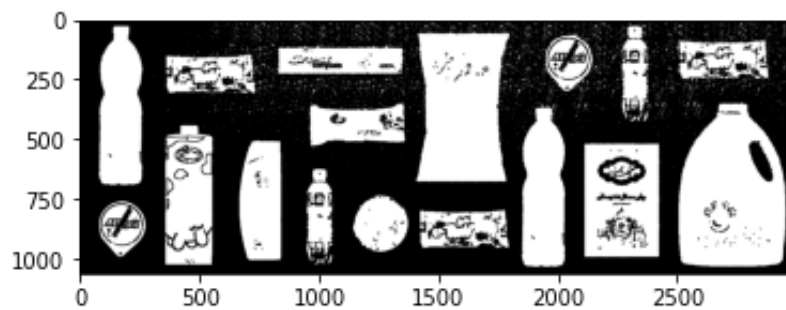


سپس آن را به gray scale تبدیل میکنیم

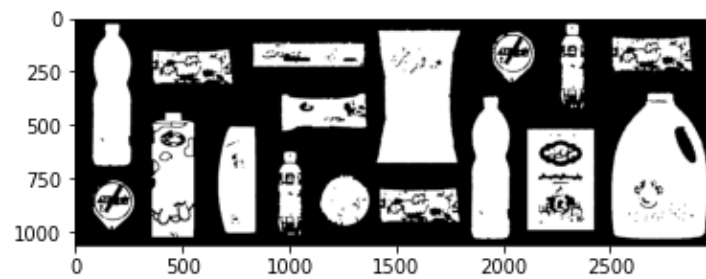


سپس با توجه به اندازه پیکسل ها، پیکسل های بالا 40 را به 255 و پیکسل های پایین 40 را به 0 تغییر میدهم تا عکس به حالت باینری در بیاید.

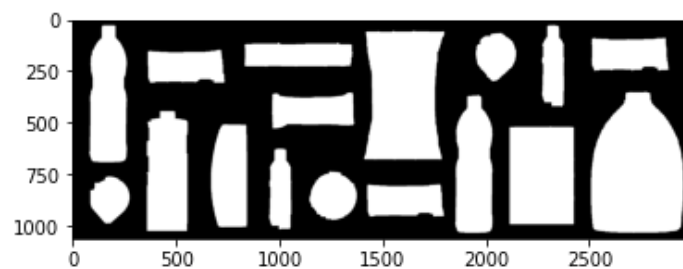
```
img_BW = img_binary.copy()
for i in range(img_binary.shape[0]):
    for j in range(img_binary.shape[1]):
        if img_BW[i][j] >= 40:
            img_BW[i][j] = 255
        elif img_BW[i][j] <= 40:
            img_BW[i][j] = 0
```



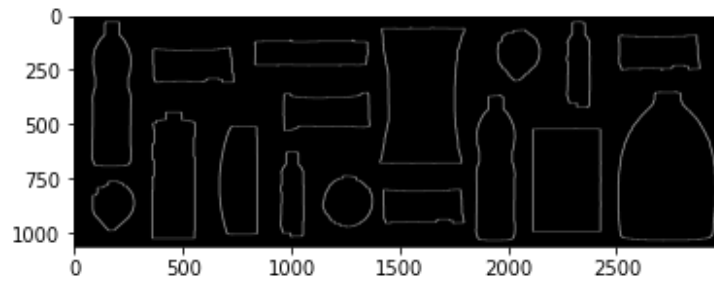
سپس بر روی نتیجه بدست آمده یک بار عمل opening را انجام می‌دهیم.



و سپس عمل closing را با اندازه کرنل (50,50) اعمال می‌کنیم.



در نهایت با اندازه کرنل (3,3) بر روی آن عمل گرادیان گیری را انجام می‌دهیم



حال با استفاده از دستور زیر تعداد اجسام موجود در تصویر را بدست میاوریم.

```
1 contours, hierarchy = cv2.findContours(closing1, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
2 print('the total number of objects in cart are : ', len(contours))
```

the total number of objects in cart are : 17

(b)

برای این قسمت تابع زیر را تعریف میکنیم:

```
1 def templateMatching (img_rgb_,template, threshold= 0.52) :
2     number =[]
3     for tem in template:
4         #run the templaee matching
5         img_gray_noisy = cv2.cvtColor(img_rgb_, cv2.COLOR_BGR2GRAY)
6         res = cv2.matchTemplate(img_gray_noisy,tem,cv2.TM_CCOEFF_NORMED)
7         loc = np.where( res >= threshold)
8         f = set()
9         #mark the corresponding location(s)
10        w, h = tem.shape
11        for pt in zip(*loc[::-1]):
12            cv2.rectangle(img_rgb_, pt, (pt[0] + h, pt[1] + w), (0, 255, 0), 5)
13            sensitivity = 150
14            f.add((round(pt[0]/sensitivity), round(pt[1]/sensitivity)))
15        number.append(len(f))
16    return img_rgb_ , number
```

```
1 objects={0: 'Tiny cake', 1: 'Maxi', 2: 'Makenzi', 3: 'Pepci', 4: 'Tea', 5: 'Shampoo', 6: 'Milk',
2          7: 'Oil', 8: 'cake', 9: 'Peril', 10: 'Toothpaste', 11: 'Chips' }
3 price=[5000, 6000, 40000, 6500, 144000, 139000,30000, 108000, 8000, 83000, 50000, 36000]
```

در این تابع ابتدا یک عکس اصلی (عکس کارت 1) و عکس template را به تابع میدهیم.

- در این قسمت عکس های template همان عکس های اجسام داخل تصویر هستند که هر کدام از تصویر اصلی جدا شده و به صورت یک تصویر جدید درآمده اند به عنوان مثال:



...9

سپس با استفاده از تابع `matchTemplate` به دنبال تصویر `template` در تصویر اصلی میگردیم. در صورتی که تصویری وجود داشته باشد با استفاده از مختصات بدست آمده و مختصات تصویر `template` بر دور تصویر تشخیص داده شده یک خط سبز رنگ رسم میشود.

همچنین تعداد اجسام تشخیص داده شده نیز از طریق خط 14 بدست می آید.

حال برای تعداد کیک های کوچک داریم :

```
img_tiny_cake= cv2.imread('tiny cake.png')
img_tiny_cake=cv2.cvtColor(img_tiny_cake,cv2.COLOR_BGR2GRAY)
img_rgb_0, count = templateMatching(img1.copy(), [img_tiny_cake], threshold=0.72)
print('The number of {} are: {}'.format(objects[0], count[0]))
plt.figure(figsize=(15, 10))
plt.imshow(img_rgb_0)
```

The number of Tiny cake are: 3
<matplotlib.image.AxesImage at 0x7f1e7552dd90>



(c)

با استفاده از تعداد بدست آمده و قیمت آن ها که در قسمت b آورده شد، قیمت اجسام را محاسبه میکنیم.

```
The number of Milk are: 1
The number of Pepci are: 2
total price is : 43000
<matplotlib.image.AxesImage at 0x7ff8fbc310>
```



13


```
1 ing_nakenzi=cv2.imread('nakenzi.png')
2 ing_nakenzi=cv2.cvtColor(ing_nakenzi,cv2.COLOR_BGR2GRAY)
3 ing_tea= cv2.imread('tea.png')
4 ing_tea=cv2.cvtColor(ing_tea,cv2.COLOR_BGR2GRAY)
5 ing_champou= cv2.imread('champou.png')
6 ing_champou=cv2.cvtColor(ing_champou,cv2.COLOR_BGR2GRAY)
7 ing_oil= cv2.imread('oil.png')
8 ing_oil=cv2.cvtColor(ing_oil,cv2.COLOR_BGR2GRAY)
9 ing_peril= cv2.imread('peril.png')
10 ing_peril=cv2.cvtColor(ing_peril,cv2.COLOR_BGR2GRAY)
11 ing_toothpaste= cv2.imread('toothpaste.png')
12 ing_toothpaste=cv2.cvtColor(ing_toothpaste,cv2.COLOR_BGR2GRAY)
13 ing_chips= cv2.imread('chips.png')
14 ing_chips=cv2.cvtColor(ing_chips,cv2.COLOR_BGR2GRAY)
15
16 ing_rgb_0, count = templateMatching(ing1.copy(), [ing_tiny_cake,ing_naxi,ing_nakenzi, ing_pepci, ing_tea, ing_shampo,ing_milk,ing_oil, ing_cake, ing_peril, ing_toothpaste,ing_chips ], threshold=0.72)
17 print('The number of {} are: {}'.format(objects[0], count[0]))
18 print('The number of {} are: {}'.format(objects[1], count[1]))
19 print('The number of {} are: {}'.format(objects[2], count[2]))
20 print('The number of {} are: {}'.format(objects[3], count[3]))
21 print('The number of {} are: {}'.format(objects[4], count[4]))
22 print('The number of {} are: {}'.format(objects[5], count[5]))
23 print('The number of {} are: {}'.format(objects[6], count[6]))
24 print('The number of {} are: {}'.format(objects[7], count[7]))
25 print('The number of {} are: {}'.format(objects[8], count[8]))
26 print('The number of {} are: {}'.format(objects[9], count[9]))
27 print('The number of {} are: {}'.format(objects[10], count[10]))
28 print('The number of {} are: {}'.format(objects[11], count[11]))
29
30 plt.figure(figsize=(15, 10))
31
32 print ('total number is : ', np.sum(count))
33 plt.imshow(ing_rgb_0)
```

```

The number of Tiny cake are: 3
The number of Maxi are: 1
The number of Makenzi are: 1
The number of Pepci are: 2
The number of Tea are: 2
The number of Shampoo are: 1
The number of Milk are: 1
The number of Oil are: 2
The number of cake are: 1
The number of Peril are: 1
The number of Toothpaste are: 1
The number of Chips are: 1
total number is : 17
<matplotlib.image.AxesImage at 0x7ff8bf9c950>

```

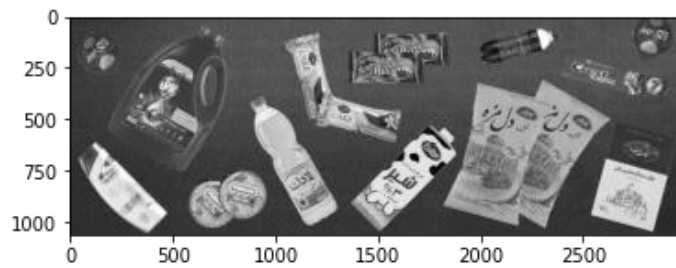


(f)

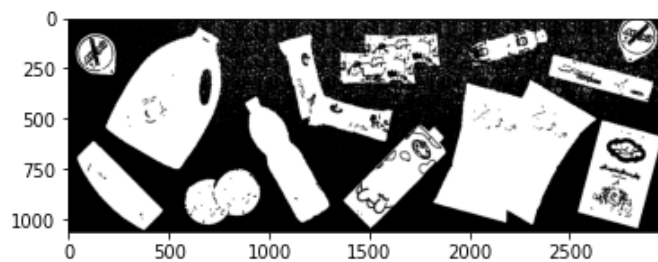
در ابتدا مانند قسمت a کارت 2 را لود میکنیم.



سپس آن را به gray scale تبدیل میکنیم.



سپس آن را به باینری تبدیل میکنیم.



و بعد از اعمال opening و closing داریم.



و در نهایت تعداد اجسام داخل تصویر برابر است با:

```
1 contours, hierarchy = cv2.findContours(closing1.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
2 print('the total number of objects in cart are : ', len(contours))
```

the total number of objects in cart are : 13

(g)

این قسمت، دشوار تر از قسمت قبلی بود. چرا که عکس های موجود در کارت 2 چرخش داشته اند. برای حل این موضوع از مبحث feature matching و homography استفاده میکنیم. برای حل این مسئله تابع زیر تعریف شده است.

با استفاده از SIFT و `SIFT.detectAndCompute` نقاط کلیدی را در دو تصویر اصلی و `template` پیدا کرده و با استفاده از `FlannBasedMatcher` به دنبال نقاط مشترک بین تصویر اصلی و `template` میگردد. حال اگر تعداد این نقاط از یک مقدار معین بیشتر باشند، آن جسم را به عنوان پیدا شده در تصویر تشخیص داده و سپس با استفاده از مختصات تصویر و جسم مورد نظر بر دور آن خط میکشد. برای شناسایی چپیس در تصویر داریم:

```
The number of Chips are: 2
<matplotlib.image.AxesImage at 0x7f74199db9d0>
```



(h)

```

1 img_cake= cv2.imread('cake.png')
2 img_cake=cv2.cvtColor(img_cake,cv2.COLOR_BGR2GRAY)
3 result_h, counts = find_objects(img2.copy(), [img_cake])
4 print('The number of {} are: {}'.format(objects[8], counts))
5 total_price= price[8] * counts
6 print ('total price is : ', total_price)
7 plt.figure(figsize=(15, 10))
8 plt.imshow(result_h)

```

The number of cake are: 2

total price is : 16000

<matplotlib.image.AxesImage at 0x7f741118ddd0>



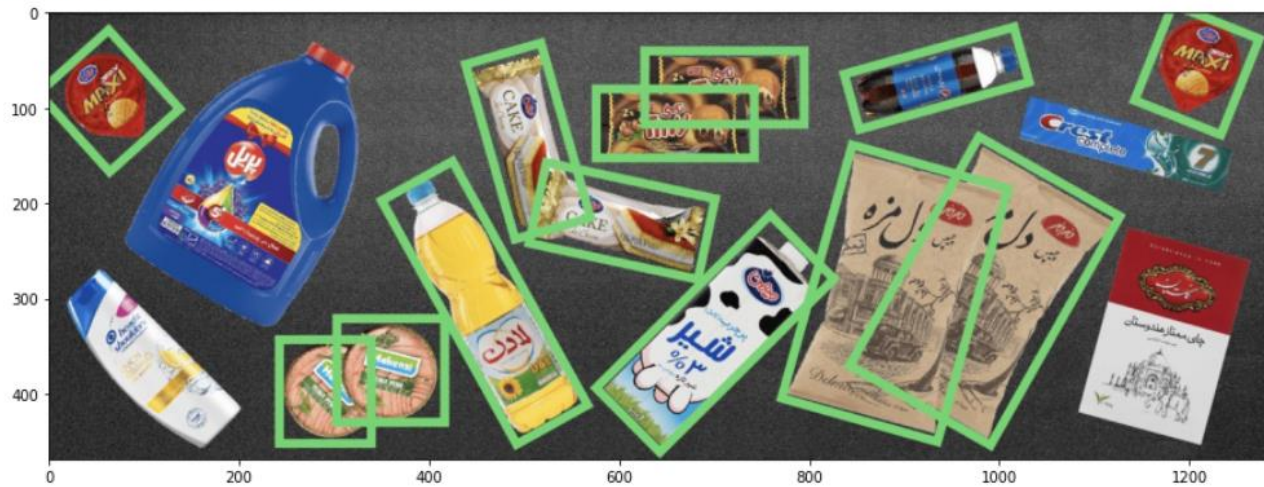
(i)

```

1 img_milk= cv2.imread('milk.png')
2 img_milk=cv2.cvtColor(img_milk,cv2.COLOR_BGR2GRAY)
3 img_tiny_cake= cv2.imread('tiny cake.png')
4 img_tiny_cake=cv2.cvtColor(img_tiny_cake,cv2.COLOR_BGR2GRAY)
5 img_pepci= cv2.imread('pepci.png')
6 img_pepci=cv2.cvtColor(img_pepci,cv2.COLOR_BGR2GRAY)
7 img_maxi= cv2.imread('maxi.png')
8 img_maxi=cv2.cvtColor(img_maxi,cv2.COLOR_BGR2GRAY)
9 img_nakenzi= cv2.imread('nakenzi.png')
10 img_nakenzi=cv2.cvtColor(img_nakenzi,cv2.COLOR_BGR2GRAY)
11 img_oil= cv2.imread('oil.png')
12 img_oil=cv2.cvtColor(img_oil,cv2.COLOR_BGR2GRAY)
13 img_chips= cv2.imread('chips.png')
14 img_chips=cv2.cvtColor(img_chips,cv2.COLOR_BGR2GRAY)
15
16
17 result_i, counts = find_objects(img2.copy(), [img_tiny_cake, img_maxi, img_nakenzi, img_pepci, img_milk, img_oil, img_cake, img_chips ])
18 print('The number of {} are: {}'.format(objects[0], counts[0]))
19 print('The number of {} are: {}'.format(objects[1], counts[1]))
20 print('The number of {} are: {}'.format(objects[2], counts[2]))
21 print('The number of {} are: {}'.format(objects[3], counts[3]))
22 print('The number of {} are: {}'.format(objects[6], counts[4]))
23 print('The number of {} are: {}'.format(objects[7], counts[5]))
24 print('The number of {} are: {}'.format(objects[8], counts[6]))
25 print('The number of {} are: {}'.format(objects[11], counts[7]))
26 total_price= price[0] * counts[0] + price[1] * counts[1] + price[2] * counts[2] + price[3] * counts[3] + price[6] * counts[4] + price[7] * counts[5] + price[8] * counts[6] + price[11] * counts[7]
27 print ('total price is : ', total_price)
28 plt.figure(figsize=(15, 10))
29 plt.imshow(result_i)

```


The number of Tiny cake are: 2
 The number of Maxi are: 2
 The number of Makenzi are: 2
 The number of Pepci are: 1
 The number of Milk are: 1
 The number of Oil are: 1
 The number of cake are: 2
 The number of Chips are: 2
 total price is : 334500
 <matplotlib.image.AxesImage at 0x7f74110aab50>




```

1 img_tea= cv2.imread('tea.png')
2 img_tea=cv2.cvtColor(img_tea,cv2.COLOR_BGR2GRAY)
3 img_shampo= cv2.imread('shampo.png')
4 img_shampo=cv2.cvtColor(img_shampo,cv2.COLOR_BGR2GRAY)
5 img_peril= cv2.imread('peril.png')
6 img_peril=cv2.cvtColor(img_peril,cv2.COLOR_BGR2GRAY)
7 img_toothpaste= cv2.imread('toothpaste.png')
8 img_toothpaste=cv2.cvtColor(img_toothpaste,cv2.COLOR_BGR2GRAY)
9
10 result_j, counts = find_objects(img2.copy(), [img_tiny_cake,img_maxi,img_makenzi, img_pepci, img_tea, img_shampo,img_milk,img_oil, img_cake, img_peril, img_toothpaste,img_chips ])
11 print('The number of {} are: {}'.format(objects[0], counts[0]))
12 print('The number of {} are: {}'.format(objects[1], counts[1]))
13 print('The number of {} are: {}'.format(objects[2], counts[2]))
14 print('The number of {} are: {}'.format(objects[3], counts[3]))
15 print('The number of {} are: {}'.format(objects[4], counts[4]))
16 print('The number of {} are: {}'.format(objects[5], counts[5]))
17 print('The number of {} are: {}'.format(objects[6], counts[6]))
18 print('The number of {} are: {}'.format(objects[7], counts[7]))
19 print('The number of {} are: {}'.format(objects[8], counts[8]))
20 print('The number of {} are: {}'.format(objects[9], counts[9]))
21 print('The number of {} are: {}'.format(objects[10], counts[10]))
22 print('The number of {} are: {}'.format(objects[11], counts[11]))
23
24 total_price= price[0] * counts[0] + price[1] * counts[1] + price[2] * counts[2] + price[3] * counts[3] + price[4] * counts[4] + price[5] * counts[5] + price[6] * counts[6] + price[7] * counts[7] + price[8] * counts[8] + price[9] * counts[9] + price[10] * counts[10] + price[11] * counts[11]
25 print ('total price is : ', total_price)
26 plt.figure(figsize=(15, 10))
27 plt.imshow(result_j)
  
```


(b)

بله میشود.

با استفاده از عمل **extensivity** میتوان پیکسل های یک عکس را به اندازه **m** به سمت راست انتقال داد. به صورتی که برای انتقال پیکسل ها از مجموعه پایین استفاده میشود که مرکز آن سمت چپ ترین خانه میباشد و زمانی که مرکز مجموعه **b** بر روی هر پیکسل از تصویر قرار میگیرد، مقدار آن را به اندازه **m** به سمت راست انتقال میدهد.

مرکز									
------	--	--	--	--	--	--	--	--	---

(c)

در **opening** و **closing** زمانی که بیشمار بار این دو عمل را بر روی عکسی انجام دهیم، در هر کدام عکس به تصویری مشخص همگرا میشود و دیگر تغییر نمیکند. در این تصویر اندازه کرنل مشخص شده در همه جای تصویر به خوبی جایی میگیرد.

(d)

عنصر ساختاری مورد استفاده در **hit and miss**، گسترش جزئی به نوعی است که برای **erosion** معرفی شده است، به این صورت که می تواند شامل پیکسل های پیش زمینه و پس زمینه باشد، نه فقط پیکسل های پیش زمینه، یعنی هم یک و هم صفر. توجه داشته باشید که نوع ساده تر عنصر ساختاری که با **erosion** استفاده می شود، اغلب حاوی هر دو یک و صفر نیز به تصویر کشیده می شود، اما در این مورد صفرها واقعاً به معنای «**don't care**» هستند و فقط برای پر کردن عنصر ساختاری استفاده می شوند. به یک هسته به شکل مناسب، معمولاً مربع. در تمام تصاویر ما، برای جلوگیری از سردرگمی، این «**don't care**» به صورت خالی در هسته نشان داده شده است.

عملیات **hit and miss** تقریباً به همان روش دیگر عملگرهای مورفولوژیکی انجام می شود، با ترجمه مبدا عنصر ساختار به تمام نقاط تصویر، و سپس مقایسه عنصر ساختار با پیکسل های تصویر زیرین. اگر پیکسل های پیش زمینه و پس زمینه در عنصر ساختار دقیقاً با پیکسل های پیش زمینه و پس زمینه تصویر مطابقت داشته باشند، پیکسل زیر مبدا عنصر ساختار بر روی رنگ پیش زمینه تنظیم می شود. اگر مطابقت نداشته باشد، آن پیکسل روی رنگ پس زمینه تنظیم می شود.

(e)

یک پیکسل که در داخل ناحیه نشان داده شده در تصویر قرار دارد. ابتدا تصویر حاوی پیکسل منفرد را با استفاده از یک عنصر ساختاری، dilate می کنیمدر نتیجه برای جلوگیری از عبور ناحیه در حال رشد از مرز، آن را AND می کنیم که منفی مرز است. با dilate کردن تصویر حاصل، AND کردن این تصویر با مرز معکوس منجر به تکرار این دو مرحله تا همگرایی، بازده و در نهایت OR کردن این تصویر با مرز اولیه، نتیجه نهایی را به دست می دهد.