تمرین ۹

سینا اسکندری ۹۷۵۲۱۰۵۴

۱-

ابتدا padding را اضافه می کنیم.

10	10	10	10	20	10	10	20	10	10
10	10	10	10	20	10	10	20	10	10
10	10	20	20	20	20	20	20	10	10
10	10	10	10	20	10	20	20	10	10
10	10	10	20	30	10	20	30	10	10
10	10	10	30	10	10	30	10	20	20
20	20	10	30	10	30	20	20	10	10
20	20	20	20	20	20	10	20	10	10
20	20	20	30	20	10	30	10	30	30
20	20	20	30	20	10	30	10	30	30

سپس با استفاده از عنصر ساختاری برای سایش، min و برای گسترش، max را برمیداریم مثلا در نقطه هایلایت شده(زرد) همسایه هایی که در عنصر ساختاری مشخص شده اند(هایلایت سبز) بررسی می کنیم که مقدار این پیکسل در سایش برابر ۱۰ و در گسترش ۳۰ می شود.

10	10	10	10	20	10	10	20	10	10
10	10	10	10	20	10	10	20	10	10
10	10	20	20	20	20	20	20	10	10
10	10	10	10	20	10	20	20	10	10
10	10	10	20	30	10	20	30	10	10
10	10	10	30	10	10	30	10	20	20
20	20	10	30	10	30	20	20	10	10
20	20	20	20	20	20	10	20	10	10
20	20	20	30	20	10	30	10	30	30
20	20	20	30	20	10	30	10	30	30

نتیجه سایش:

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	20	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
20	20	20	20	10	10	10	10

نتیجه گسترش:

10	10	20	20	20	20	20	20
10	10	20	20	20	20	20	20
20	20	20	20	30	20	20	20
10	10	20	30	30	20	30	30
20	20	30	30	30	30	30	30
20	30	30	30	30	30	30	20
20	30	30	30	30	30	30	20
20	20	20	30	20	20	30	20

۲-

میتوان از عنصر های زیر استفاده کرد که مرز ها را بدست آورد

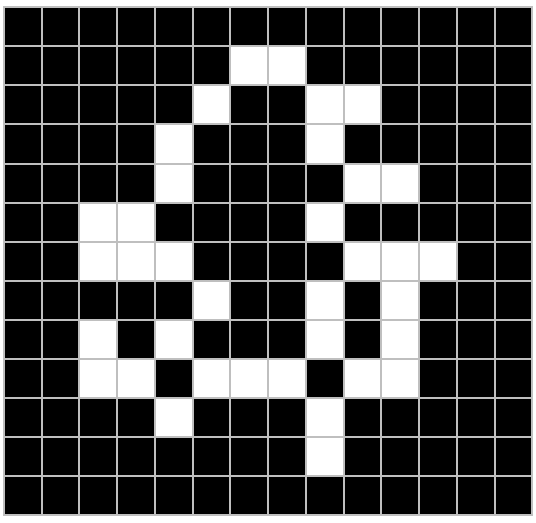
0	0	0
0	1	-1
0	0	0

0	0	0
-1	1	0
0	0	0

0	-1	0
0	1	0
0	0	0

0	0	0
0	1	0
0	-1	0

که نتیجه hit or miss به این صورت می شود:



۳-

الف)

طبق این فرمول ها عمل میکنیم.

$$S(A) = \bigcup_{k=0}^{K} S_k(A)$$

$$S_k(A) = (A \ominus kB) - (A \ominus kB) \circ B$$

$$A \ominus kB = \big((A \ominus B) \ominus B\big) \ominus \cdots)$$

$$K = max\{k | (A \ominus kB) \neq \emptyset\}$$

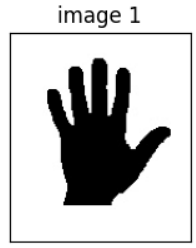
$$A = \bigcup_{k=0}^{K} S_k(A) \oplus kB$$

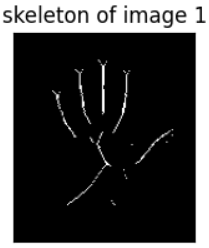
ابتدا تصویر را باینری می کنیم و سپس جای رنگ ها را برعکس می کنیم که این فرمول ها درست اعمال شوند.

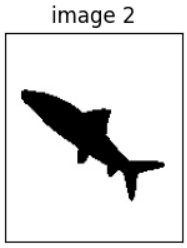
```python
def threshold(img):
    ret, im = cv2.threshold(img, 127, 1, 0)
    return im

def invert_color(img):
    im = np.zeros_like(img)
    im[img == 0] = 1
    return im
```

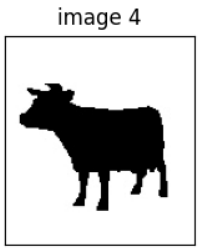
```python
res = image.copy()
params = []

#Write your code here
res = invert_color(threshold(res))

skeleton = np.zeros_like(res)
element = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
k = 0

while cv2.countNonZero(res):
    open = cv2.morphologyEx(res, cv2.MORPH_OPEN, element)
    s_k = cv2.subtract(res, open)
    params.append((k, s_k))
    erode = cv2.erode(res, element)
    skeleton = np.bitwise_or(skeleton, s_k)
    res = erode.copy()
    k += 1

return skeleton, params
```

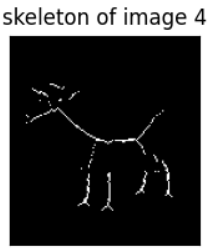
در هر مرحله عملگر open تصویر و عنصر ساختاری را حساب میکنیم و از تصویر کم میکنیم و k و S_k را به params اضافه می کنیم که برای قسمت بعد لازم است.نتیجه S_k را با اسکلتی که مراحل قبل تشکیل شده or می کنیم که معادل خط اول فرمول است.
در آخر هر مرحله نیز تصویر را با عنصر ساختاری erode می کنیم که $A \ominus kB$ بدست آید. این کار را تا زمانی انجام میدهیم که نتیجه erode حداقل ۱ پیکسل داشته باشد که مخالف صفر باشد.







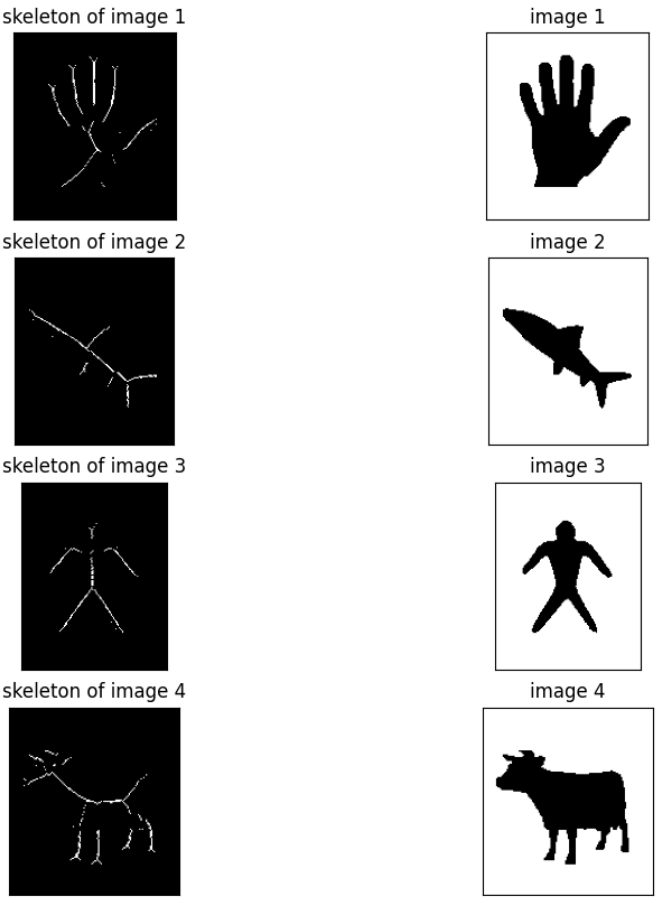




ب)

برای این قسمت کافی است از خط آخر فرمول استفاده کنیم و از آنجا که k و S_k را از قسمت قبل ذخیره کرده ایم، باید هر S_k را k بار با عنصر ساختاری dilate کنیم و با نتیجه or کنیم.

```
# res = image.copy()
res = np.zeros_like(image)

#Write your code here

element = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))

for k, s_k in params:
    dilation = cv2.dilate(s_k, element, iterations=k)
    res = np.bitwise_or(res, dilation)
res = invert_color(res)
return res
```

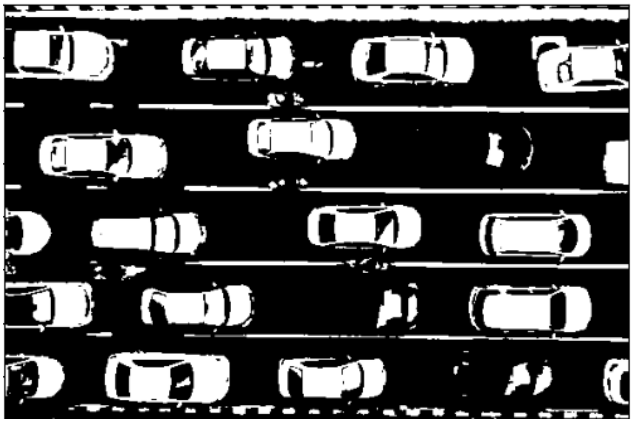


۴-

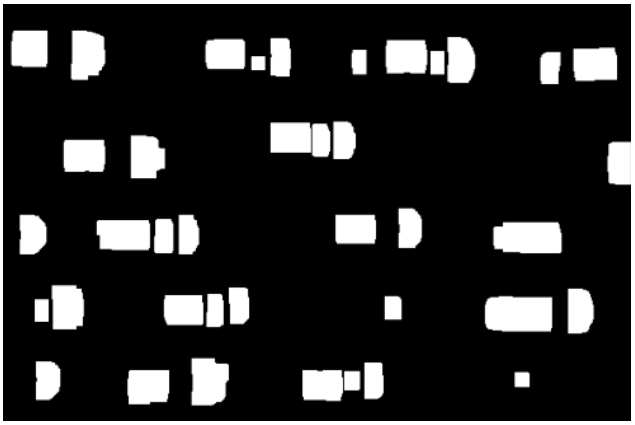
الف)

ابتدا با استفاده از فیلتر گوسی تصویر را smooth میکنیم و سپس با استفاده از روش otsu تصویر را باینری می کنیم.

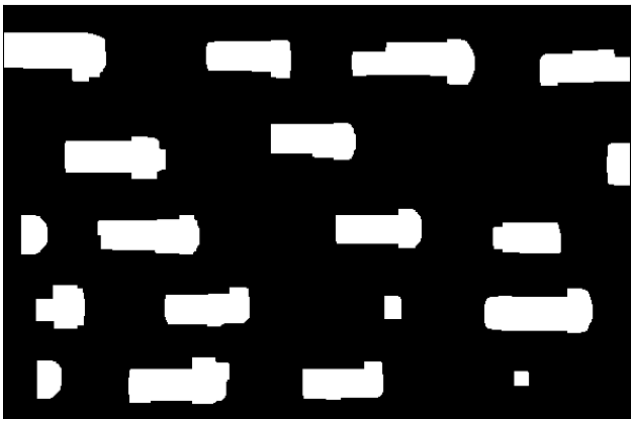
نتیجه باینری:



سپس تصویر را open میکنیم



برای اینکه قسمت های سفید به هم بچسبند close میکنیم.

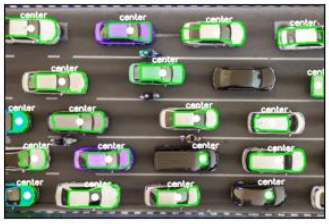


مراحل بعدی را طبق لینک اول پیش می بریم.

```
#Write your code here
gray = cv2.cvtColor(result, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (3, 3), 0)
ret, thresh = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
opening_element = cv2.getStructuringElement(cv2.MORPH_RECT, (13, 13))
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, opening_element)
closing_element = cv2.getStructuringElement(cv2.MORPH_RECT, (27, 27))
closing = cv2.morphologyEx(opening, cv2.MORPH_CLOSE, closing_element)

cnts = cv2.findContours(closing.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)

for c in cnts:
# compute the center of the contour
    cars_num += 1
    M = cv2.moments(c)
    cX = int(M["m10"] / M["m00"])
    cY = int(M["m01"] / M["m00"])
    # draw the contour and center of the shape on the image
    cv2.drawContours(result, [c], -1, (0, 255, 0), 2)
    cv2.circle(result, (cX, cY), 7, (255, 255, 255), -1)
    cv2.putText(result, "center", (cX - 20, cY - 20),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)


return result, cars_num
```

source

cars detection



ب)

برای این قسمت ابتدا smooth میکنیم. سپس canny اجرا میکنیم چون گفته شده که نباید باینری کنیم. بقیه قسمت ها شبیه قسمت قبل است با این تفاوت که در قسمت contours فقط آن هایی که دایره هستند انتخاب می کنیم.

منابع

- لینک های معرفی شده در داک تمرین