

دانشگاه علم و صنعت

تمرین پنجم مبانی بینایی کامپیوتر

نام و نام خانوادگی:

فرناز خوش دوست آزاد

شماره دانشجویی:

99521253

نام استاد:

دکتر محمدرضا محمدی

۱. به شما تصویر q1 داده شده است. از شما میخواهیم که connected component ها را پیدا کرده و هر component را با رنگ متفاوتی برچسب بزنید. همچنین، تعداد کل component ها را روی تصویر چاپ کنید (استفاده از توابع آماده مجاز است). (سوال عملی - ۵ نمره)

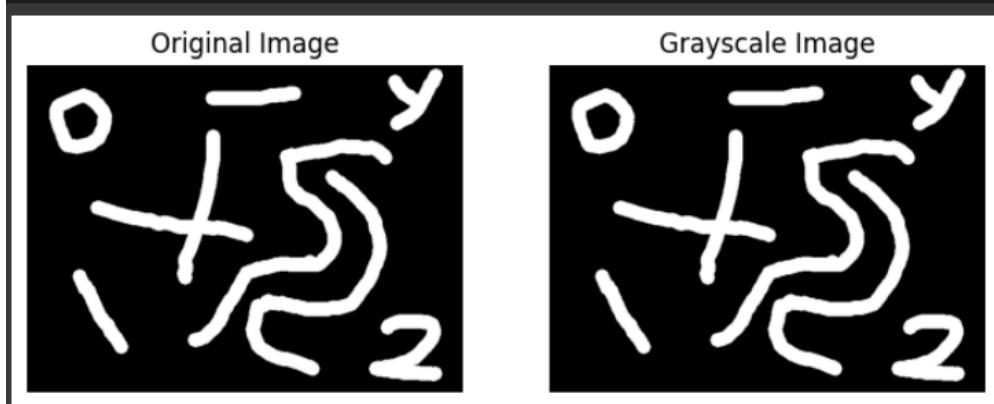
پاسخ:

همانطور که در فایل hw5_q1.ipynb نیز موجود است در ابتدا کتابخانه های مورد نظر را import کرده ام و سپس از پوشه images عکس مورد نظر را با کتابخانه PIL خوانده ام و سپس با استفاده از تصویر را به هر دو صورت رنگی و grayscale خوانده ام که واضح است که تفاوتی با یکدیگر ندارند که در تصویر زیر نیز به خوبی مشخص است:

```

2     gray_image = image.convert("L")
3     plt.figure(figsize=(8, 6))
4
5     # Original Image
6     plt.subplot(1, 2, 1)
7     plt.title("Original Image")
8     plt.imshow(image)
9     plt.axis("off")
10
11    # Grayscale Image
12    plt.subplot(1, 2, 2)
13    plt.title("Grayscale Image")
14    plt.imshow(gray_image, cmap="gray")
15    plt.axis("off")
16
17    plt.show()
18

```



سپس همانطور که در کد زیر مشخص است و کامنت گذاری شده است در ابتدا تصویر به gray تبدیل شده است، سپس تصویر را به یک آرایه ای از numpy تبدیل کرده ام و سپس threshold = 0 در نظر گرفته ام که به صورت boolean مقادیر را برابر دو کلاس باینری بر می گرداند و سپس با استفاده از

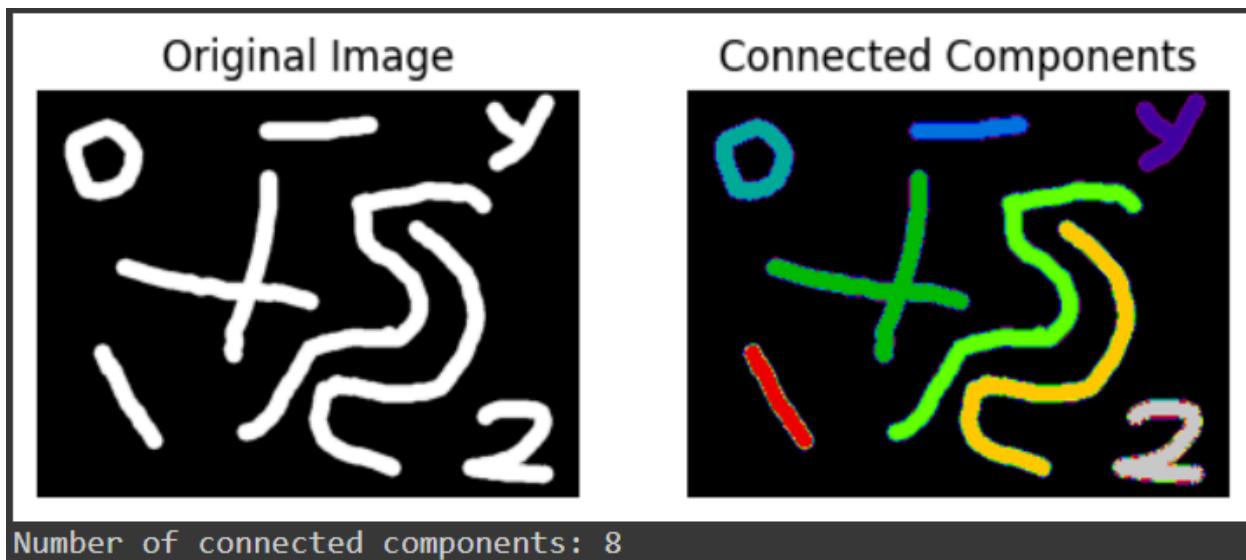
تصاویر `label_image` شده را در `binary` ذخیره کرده ام و سپس `max` آن را برگردانده ام که همان تعداد `components` ما می باشد و در آخر آن را پایین عکس های خود کرده ام.

```

1  image_gray = image.convert('L')
2  image_array = np.array(image_gray)
3
4  # Threshold the image to create a binary image
5  binary_image = image_array > 0
6  # Find connected components in the binary image
7  label_image = measure.label(binary_image)
8  # Count the number of connected components
9  num_components = np.max(label_image)

```

که نتیجه کد خود را در آخر با استفاده از `plt` نشان داده ام که به صورت زیر است:



(2)

۲. الگوریتم رشد ناحیه را برای تصویر ۹۲ پیاده‌سازی کنید. به طوری که براساس نقطه `seed` شما چهره فرد را به رنگ دلخواهتان در بیاورد. دو حالت همسایگی ۴ تایی و ۸ تایی را پیاده سازی کرده و با هم مقایسه کنید. حد آستانه‌های مختلف را تست کنید و نتایج مختلف را در گزارش کار بیاورید. از توابع آماده نباید استفاده کنید (نوتبوك ۹۲ را کامل کنید). (سوال عملی - ۱۵ نمره)

: پاسخ

چالش:

چالشی که در این سوال برخوردم، این بود که در ابتدا با استفاده از `cv2 threshold` به رسم اشکال با های مختلف پرداختم، اما همانطور که در انتهای نوتبوک نیز مبرهن است، خطوطی آبی بدون دلیل در نقاط x و y برای `seed_point` نیز در آن رسم شده است که دلیل آن را هنوز پیدا نکرده ام، اما با استفاده از `plt` توانستم تصاویر بدون خطوط درستی را تولید کنم.

شرح کد:

در ابتدا برای پیدا کردن جای حدودی `seed_point` کدی ردم تا جای خودودی آن با شکل لوزی آبی دیده شود و سپس به سراغ فانکشن خواسته شده رفتم.

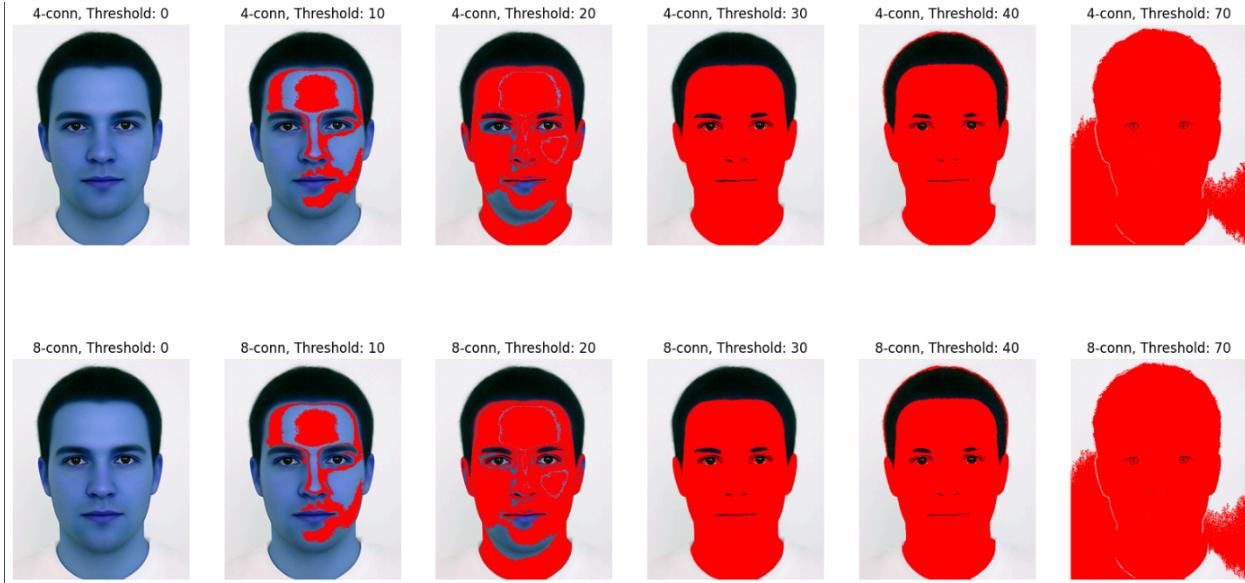
در کد زده شده در فانکشن `segment1` در ابتدا کپی ای از تصویر ورودی می گیرم و سپس `offset` مربوط به `seed_point` و `connectivity` 4 و 8 را نوشتم. و سپس نقاط یا نقطه‌ی `seed_point` را به استک اضافه می کنم و تا زمانی که استک من خالی نشود لوپی را در آن تکرار می کنم.

در این لوپ با توجه به ورودی در ابتدا `offset` های خود را از میان دو کلاس انتخاب می کنم. سپس با توجه به `seed_point` نقاط کناری آن را در صورتی که شرط گذاشته شده را برآورده کند در صورت `unvisited` بودن به داخل استک اضافه می کنم و دوباره همین روند را ادامه می دهم که کد آن نیز به صورت زیر می باشد. شرط نوشته شده بین صورت است که سه کانال نقطه‌ی `seed_point` را می گیرم و سپس اختلاف آن را با سه کانال نقطه‌ی مورد نظر حساب می کنم و در صورتی که میانگین آنها از `threshold` کمتر باشد، آن را به نقطه‌ی قرمز رنگ تبدیل می کنم و همان نقطه را `visited` در نظر می گیرم و به داخل استک خود نیز اضافه می کنم.

حالات های مختلف:

در داخل نوتبوک نیز موارد مختلف نشان داده شده اند.

برای مثال در ابتدا چند نقطه‌ی `Seed_point` را اضافه کرده ام که نتیجه‌ی آن نیز به صورت زیر شد:



نقاط گرفته شده اولیه در تصاویر بالا نیز به شرح زیر است که سه نقطه می باشند.

```

seed_point = [(200, 200), (170, 140), (120, 160)]
threshold =[0, 10, 20, 30, 40, 70]

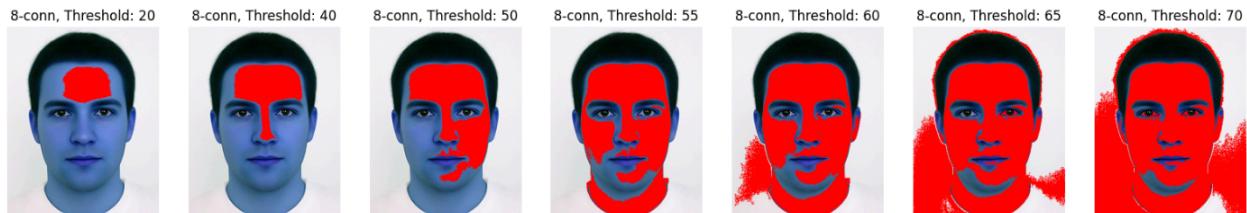
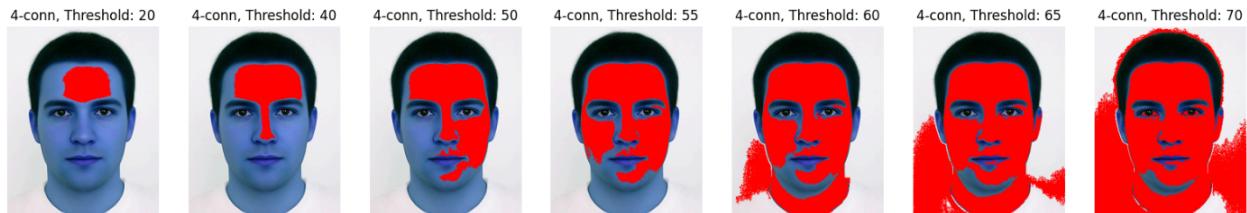
fig, axes = plt.subplots(2, len(threshold), figsize=(20, 10))
for i in range(len(threshold)):

    segmented_image_4 = segment1(image,seed_point,threshold[i])
    segmented_image_8 = segment1(image,seed_point,threshold[i],1)

    axes[0, i].imshow(segmented_image_4)
    axes[0, i].set_title(f'4-conn, Threshold: {threshold[i]}')
    axes[0, i].axis('off')
    axes[1, i].imshow(segmented_image_8)
    axes[1, i].set_title(f'8-conn, Threshold: {threshold[i]}')
    axes[1, i].axis('off')

```

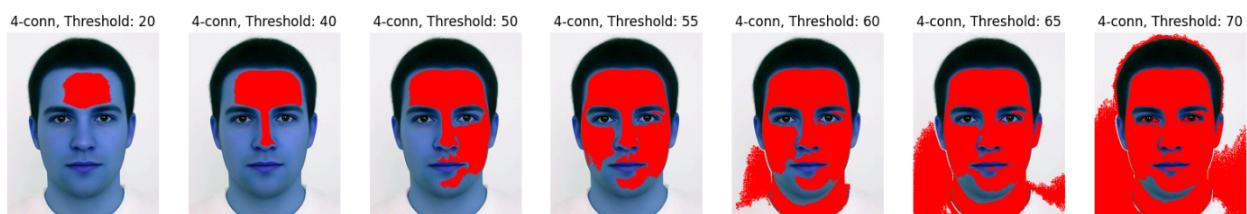
حال تنها برای نقطه‌ی (150 و 150) تصاویر به دست آمده را رسم می کنم که نتیجه به صورت زیر است:



می بینیم که در تصاویر بالا قسمت زیادی از تصویر پشت زمینه شخص نیز به عنوان تصویر فرد تشخیص داده می شود، لذا نقطه‌ی دیگری را برای نتیجه‌ی دیگر انتخاب کردم.

در تصویر اولی که گذاشتم دیده می شود که تصویر 4 connectivity و 8 connectivity فرقی ندارند اما در تصویر بالا برای $\text{threshold} = 55$ می بینیم که به در صورت استفاده از 8 گردن فرد نیز تشخیص داده می شود. که به دلیل جستجوی منحصر به فرد آن است.

حال می بینیم با انتخاب نقطه‌ی (200 و 200) تصویر زیر به دست آمد.



همانطور که دیده می شود با انتخاب نقطه ای دیگر نتیجه ای بهتری نیز به دست آوردهیم. که از میان تصاویر بالا تصویر وسط بهترین threshold ما می باشد.

در ادامه نوتبوک نیز چالشی که به آن برخورده بودم را آورده ام که هنوز دلیلی برای آن پیدا نکرده ام.

(3)

۳. به صورت تصادفی یک تصویر ۵ در ۵ تولید کنید (یک کاناله و مقادیر آن در بازه ۱ تا ۱۵ باشند). سپس روی کاغذ الگوریتم Otsu را برای سطح آستانه‌های ۶ و ۱۰ روی آن اجرا کنید. سطح آستانه بهتر کدام است؟ (سوال تئوری -۱۰ نمره)

پاسخ:

12	6	7	3	4
10	5	4	13	5
11	10	4	11	11
14	1	7	9	9
8	11	14	9	8

در ابتدا هیستوگرام تصویر را حساب می کنیم که به صورت زیر است

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	3	2	1	2	2	3	2	4	1	1	2	0

حال با سطح آستانه ۶ دو کلاس خواهیم داشت که در کلاس اول تنها یک تا ۵ و در کلاس دیگر مقادیر ۶ تا ۱۵ قرار دارند. حال باید با توجه به فرمول زیر مقادیر را حساب کنیم که داریم:

$$\text{برای سطح آستانه} = 6$$

$$\begin{aligned}
 & \text{و میانسیز درجه دار} \\
 & 1 \times 1 + 0 \times V + \frac{3 \times 1 + 4 \times 1 + 1 \times 1}{V} + 1 \times 0 = \frac{14}{V} \\
 & 8 \left\{ 4 \times 1 + V \times 1 + 1 \times 1 + 4 \times 1 + 1 \times 1 + 11 \times 1 + 12 \times 1 + 13 \times 1 \right. \\
 & \quad \left. + 14 \times 1 + 10 \times 0 \right) \div 18 = \frac{18}{18} = 10 \\
 & \text{روش الگوریتم: } \frac{V}{20} \left(\left(1 - \frac{14}{V} \right)^2 + \left(3 - \frac{14}{V} \right)^2 + 3 \times \left(4 - \frac{14}{V} \right)^2 + 1 \times \left(0 - \frac{14}{V} \right)^2 \right) + \\
 & \text{سطح آستانه: } \frac{18}{20} \left(\left(4 - 10 \right)^2 + 2 \left(V - 10 \right)^2 + 3 \times \left(1 - 10 \right)^2 + 3 \times \left(9 - 10 \right)^2 + 2 \left(10 - 10 \right)^2 \right. \\
 & \quad \left. + 4 \left(11 - 10 \right)^2 + 1 \left(12 - 10 \right)^2 + \left(13 - 10 \right)^2 + 2 \left(14 - 10 \right)^2 \right) = \\
 & \frac{V}{20} \times 13 / 20 V + \frac{18}{20} \times 94 \approx 11 / 14
 \end{aligned}$$

و برای سطح آستانه = 10

$$\begin{aligned}
 & \text{و میانسیز درجه دار} \\
 & (1 \times 1 + 2 \times 0 + 3 \times 1 + 4 \times 1 + 0 \times 1 + 4 \times 1 + V \times 1 + 1 \times 1 + 4 \times 1) \div 10 = \frac{14}{10} \\
 & 8 \left\{ 10 \times 1 + 11 \times 1 + 12 \times 1 + 13 \times 1 + 14 \times 1 \right\} \div 10 = \frac{110}{10} = 11 V \\
 & \text{روش الگوریتم: } \frac{10}{20} \left(\left(1 - \frac{14}{10} \right)^2 + \left(2 - \frac{14}{10} \right)^2 + \left(3 - \frac{14}{10} \right)^2 + 2 \left(0 - \frac{14}{10} \right)^2 + \left(4 - \frac{14}{10} \right)^2 \right. \\
 & \quad \left. + 2 \left(V - \frac{14}{10} \right)^2 + 2 \left(1 - \frac{14}{10} \right)^2 + 3 \left(9 - \frac{14}{10} \right)^2 + \right. \\
 & \quad \left. \frac{10}{20} \left(2(10 - 11, V)^2 + 4(11 - 11, V)^2 + (12 - 11, V)^2 + (13 - 11, V)^2 + 2(14 - 11, V)^2 \right) = \right. \\
 & \quad \left. \frac{10}{20} (14, 91) + \frac{10}{20} (20, 10) = 50, 94 + 1, 04 \approx \boxed{51, 94} \right)
 \end{aligned}$$

با توجه به جواب های به دست آمده می بینیم که برای مقدار دهی رندوم من سطح آستانه 10 جواب بهتری به ما می دهد و تقریبا 14 واحد کمتر از زمانی است که سطح آستانه را 6 قرار می دهیم که برای مقادیر رندوم دیگر ممکن است بر عکس و یا حتی این دو مقدار برابر هم شوند. من برای به دست آوردن این فرمول داخل پی دی اف استفاده کردم که به شرح زیر است:

- یک الگوریتم تعیین سطح مقدار آستانه بر حسب مشخصه‌های آماری است
- خلاصه الگوریتم این است که سطح آستانه‌ای را انتخاب کنیم که واریانس بین پیکسل‌های هر کلاس کمینه شود

$$\sigma_w^2 = w_1 \sigma_1^2 + w_2 \sigma_2^2$$

- w_i تعداد پیکسل‌های کلاس i ، و σ_i^2 واریانس پیکسل‌های آن کلاس است

(4)

۴. صادق به تازگی با آستانه‌گذاری افقی آشنا شده است و بر روی تصویر کتاب خود، این عملیات را با پنج حالت مختلف اعمال کرده است. اما حالا نمی‌داند که کدام تصویر حاصل مربوط به کدام ترکیب آرگومان‌ها است. شما با مراجعه به تصویرهای [1-5]، آرگومان‌های C و q4_1-[5] و blockSize و thresholdType برای هر تصویر [1-5] را مشخص کنید و دلیل خود را توضیح دهید. (سوال تئوری ۱۰- نمره)
مقادیر ممکن برای هر پارامتر به شرح زیر است:

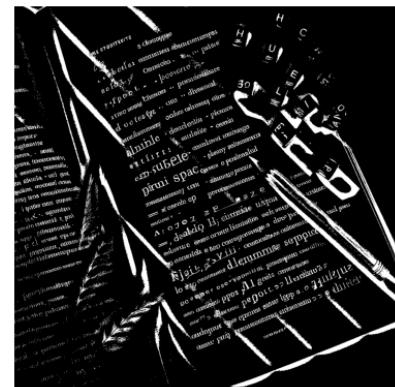
C: [5, 30]

blockSize: [21, 41]

thresholdType: [THRESH_BINARY, THRESH_BINARY_INV]







پاسخ:

می دانیم که آستانه گذاری افقی با کمک تابع `cv2.adaptive Threshold` صورت می گیرد که برخی ورودی های آن عبارتند از `BlockSize` و `thresholdType` و `C`. پارامتر بیانگر آن است که اگر $\text{mean} + c < \text{src}$ باشد به رنگ سفید نمایش داده شود و یا در صورت بر عکس بوده به صورت مشکی دیده می شود. پارامتر دوم نیز ابعاد پنجره ای است که برای محاسبه میانگین در نظر گرفته می شود و هر چه ابعاد بزرگتر باشد به آستانه گذاری سراسری نزدیکتر می شود. لذا باید ابعاد پنجره می متناسب برای خود را انتخاب کنیم و پارامتر آخر نیز برای شیفت دادن آستانه نسبت به میانگین پنجره می باشد.

در تصویر اول، سمت چپ پایین تصویر نوشته ها واضح نیستند و به خوبی `binary` نشده اند. بنابراین این تصویر از `blockSize` بزرگتر از 41 استفاده می کند. همچنین این تصویر نسبت به سایرین کمی روشنتر است و احتمالا `c` کوچکتری داشته است. ($C = 5$)

در تصویر دوم همه نواحی با دقت قابل قبولی `binary` شده اند، پس پنجره ای کوچکتری استفاده شده است (`block size 21`)، اما تصویر به طور کلی کمی تیره تر از حالت عادی به نظر می رسد. می توان حدس زد که $c = 30$ مقدار بیشتری داشته است و باعث شده تصویر تیره تر از حالت میانگین باشد.

در تصویر سوم نیز نواحی به خوبی binary شده اند، پس پنجره کوچکتری داشته ($\text{blockSize} = 21$) همچنین در مقایسه با تصویر دوم، به نظر می رسد که خطوط حاشیه مداد و خطوط مورب در حالت عادی باشد و پررنگ تر یا کمرنگ تر از حد معمول نیستند! پس مقدار $C = 5$ می باشد تا آستانه ما حدودا همان میانگین پنجره باقی بماند.

در تصویر چهارم مجددا ناحیه پایین چپ حالت محوشده دارد. پس پنجره‌ی بزرگتری داشته است ($\text{blockSize} = 41$)، همچنین خطوط مداد و مورب پررنگ تر از حالت عادی به نظر می رسد. به همین خاطر احتمالا مقدار C بزرگتری داشته اند. ($C = 30$).

در تصویر آخر نیز که تصویری با پس زمینه‌ی مشکی می باشد متوجه می شویم که `thresholdType` برابر با همان `THRESH_BINARY_INV` است. لذا متوجه می شویم که روشنایی بیشتر از آستانه به صفر تناظر بافته و کمتر از آستانه به 255 تبدیل می شود. حال مجددا ناحیه‌ی سمت چپ پایین تصویر نیز به خوبی binary نشده است. لذا $\text{blockSize} = 41$ و $C = 5$ می باشد.

(5)

۵. عملگر سایش و گسترش را با توجه به عنصر ساختاری داده شده، بر روی تصویر زیر اعمال کنید. (در صورت نیاز از استفاده نمایید، همچنین لنگر `anchor` در مرکز عنصر ساختاری قرار دارد). (سوال تئوری-

۱۰ نمره)

1	1	1
1	0	0
1	0	0

22	22	22	33	22	22	33	22
22	33	33	33	33	33	33	22
22	22	22	33	22	33	44	22
22	22	33	44	22	33	22	22
22	22	44	22	22	44	33	22
33	22	44	22	44	33	33	22
33	33	33	33	33	22	33	22
33	33	44	33	22	44	22	44

پاسخ:

در ابتدا `reflect` را بر روی تصویر اعمال می کنیم که داریم که نتیجه‌ی آن مانند شکل زیر می باشد:

22	22	22	22	33	22	22	33	22	22
22	22	22	22	33	22	22	33	22	22
22	22	33	33	33	33	33	33	22	22
22	22	22	22	33	22	33	44	22	22
22	22	22	33	44	22	33	22	22	22
22	22	22	44	22	22	44	33	22	22
33	33	22	44	22	44	33	33	22	22
33	33	33	33	33	33	22	33	22	22
33	33	33	44	33	22	44	22	44	44
33	33	33	44	33	22	44	22	44	44

حال برای سایش برای هر پیکسل باید از بین مربع هایی که یک هستند، مینیمم آنها را انتخاب کرده و سپس به جای هر پیکسل جایگذاری کنیم.

1	1	1
1	0	0
1	0	0

که پاسخ آن به شرح زیر است:

22	22	22	22	22	22	22	22
22	22	22	22	22	22	22	22
22	22	22	22	33	22	22	22
22	22	22	22	22	22	22	22

22	22	22	22	22	22	22	22
22	22	22	22	22	22	22	22
22	22	22	22	22	22	22	22
33	33	33	33	22	22	22	22

که این شکل حاصل از سایش توسط فیلتر بالا می باشد. حال برای گسترش باید در ابتدا فیلتر را 180 درجه دوران دهیم و سپس برای هر پیکسل از بین خانه هایی که یک می باشند، maximum را انتخاب می کنیم که فیلتر ما به صورت زیر می شود.

0	0	1
0	0	1
1	1	1

و پاسخ ما نیز پس از reflect به صورت زیر می شود:

33	33	33	33	33	33	33	33
33	33	33	33	33	44	44	44
33	33	44	44	44	44	33	22
22	44	44	44	44	44	44	33
33	44	44	44	44	44	33	33
33	44	33	44	44	33	33	33
33	44	44	44	44	44	44	44
33	44	44	44	44	44	44	44

که جدول بالا شکل حاصل از گسترش می باشد که ماکریم را برای هر یک به دست آورده ایم.

(6)

۶. نتیجه اعمال مورفولوژی زیر بر روی تصویر نشان داده شده چه خواهد بود؟ این عملیات چه پردازشی بر روی تصویر ورودی انجام می دهد؟ (سوال تئوری- ۵ نمره) $(A \ominus B_1) \cap (A^c \ominus B_2)$

B1

0	0	0
1	1	0
0	1	0

B2

0	1	1
0	0	1
0	0	0

پاسخ

A. B1

0	0	0	0						0	0	0
0	0	0	0						0	0	0
0	0	0	0						0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Ac , B2

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0					✉	0	0	0
0	0	0	0						0	0	0
0	0	0	0						0	0	0
0										0	0
0										0	0
0										0	0
0										0	0
0	0	0	0					0	0	0	0
0	0	0	0					0	0	0	0
0	0	0	0					0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

نقطه‌ی اشتراک در شکل بالا با رنگ قرمز نمایش داده شده است که نمایش می‌دهد که این فرمول گوشه‌های بالای سمت راست را نمایش می‌دهد.

۷. با استفاده از علم مورفولوژی و به کمک لینک های ۱ و ۲ خواسته های زیر را انجام دهید. (توجه شود که در این سوال، امکان استفاده از توابع کتابخانهای را دارید)

(الف) برنامه‌ای بنویسید که تعداد خودروهای موجود در تصویر car.jpg را بدست آورد. (راهنمایی: تصویر داده شده با روش‌های موجود در درس به تصویری باینری تبدیل، و بوسیله عملگرهای مورفولوژی خطوط اضافی را حذف نموده و با توابع کمکی خودروها را یافته و تعداد آنها را گزارش نمایید). (سوال عملی ۱۵- نمره)

(ب) برنامه‌ای بنویسید که تعداد گل آفتابگردان موجود در تصویر flower.jpg را محاسبه نماید. (توجه: در این بخش باید از حالت رنگی تصویر استفاده نمایید و تصویر را باینری نکنید. همچنین برای یافتن تعداد گل‌ها می‌توانید بر روی بخش دایروی گل تمرکز نمایید و با یافتن آن، تعداد گل‌ها را ببایدید) (سوال عملی- امتیازی- ۱۰ نمره)

پاسخ:

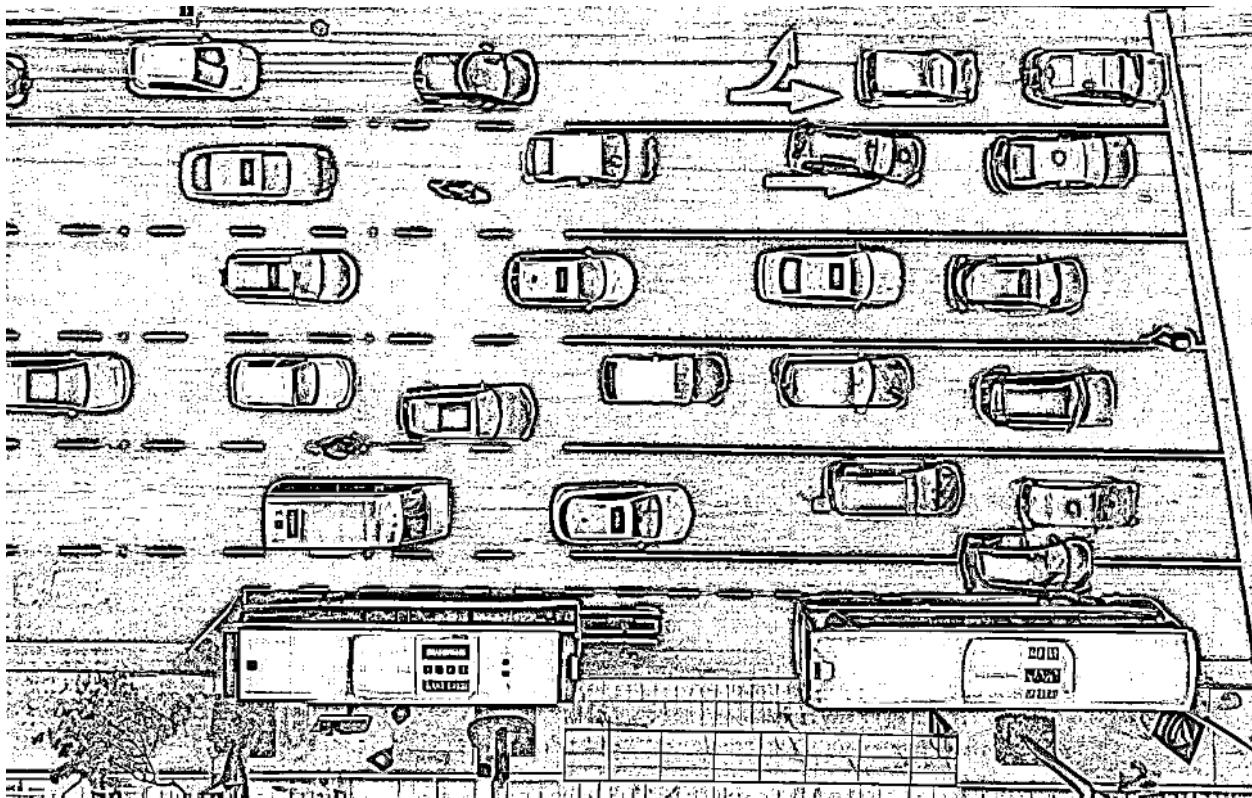
در ابتدا تصویر اصلی و باینری و gray scale آن را به نمایش می‌گذاریم:
تصویر اصلی



تصویر :gray_Scale



تصویر باینری:



حال باید به سراغ نویز برویم و آن را با استفاده از عملیات های مورفولوژی حذف کنیم که کد آن به شرح زیر است:

```
1 # delete noise:  
2 kernel = np.ones((5, 5), np.uint8)  
3 opening = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel)  
4 opening
```

حال باید contours ها را در تصویر پیدا کنیم که کد آن به صورت زیر است:

```
1 # find counters  
2 contours, _ = cv2.findContours(opening, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)  
3 contours
```

حال باید contours را بر اساس ناحیه فیلتر کرده و تعداد ماشین ها را می شماریم و مقدار دیگر را که در کد زیر است را با آزمون و خطابه دست می آوریم و نتیجه و کد در تصویر زیر آورده شده اند:

```
1 min_contour_area = 3450  
2 large_contours = [cnt for cnt in contours if cv2.contourArea(cnt) > min_contour_area]  
3  
4 num_cars = len(large_contours)  
5 print(f'Number of Cars in the image: {num_cars}')
```

Number of Cars in the image: 23

(8)

الف) بدون استفاده از توابع آماده کتابخانه‌ای و با استفاده از دانش مورفولوژی اسکلت تصاویر [5-7]_q8 را استخراج کنید. (سوال عملی- ۱۵ نمره)

ب) با ذخیره مراحل استخراج اسکلت، تصاویر اولیه را از روی اسکلت بازسازی کنید. (سوال عملی- ۱۰ نمره)

پاسخ:

با توجه به فرمولی که در اسلاید ها اشاره شده است باید به صورت زیر عمل کنیم.

$$\begin{aligned}
 S(A) &= \bigcup_{k=0}^K S_k(A) \\
 S_k(A) &= (A \ominus kB) - (A \ominus kB) \circ B \\
 A \ominus kB &= ((A \ominus B) \ominus B) \ominus \dots \\
 K &= \max\{k | (A \ominus kB) \neq \emptyset\} \\
 A &= \bigcup_{k=0}^K S_k(A) \oplus kB
 \end{aligned}$$

که کد آن به شرح زیر است:

```

1 def skeletonize(input_image):
2     skeleton = np.zeros_like(input_image, dtype=np.uint8)
3     structuring_element = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
4     intermediate_steps = []
5
6     current_image = input_image.copy()
7     while True:
8         eroded_image = cv2.erode(current_image, structuring_element)
9         opened_image = cv2.morphologyEx(eroded_image, cv2.MORPH_OPEN, structuring_element)
10        temp_image = cv2.subtract(eroded_image, opened_image)
11        skeleton = cv2.bitwise_or(skeleton, temp_image)
12        intermediate_steps.append(temp_image)
13        current_image = eroded_image.copy()
14        if cv2.countNonZero(current_image) == 0:
15            break
16
17    return skeleton, intermediate_steps
18

```

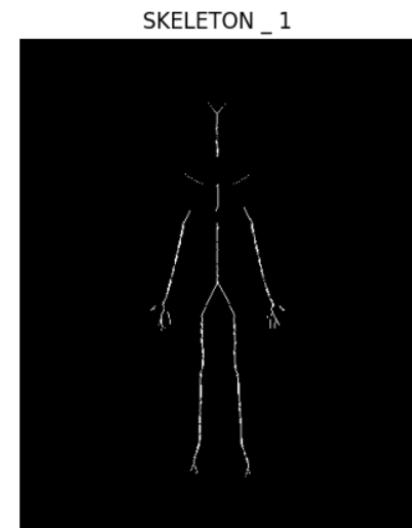
و همچنین برای در هر مرحله sk ما سیو می شود و در نهایت ما اسکلت داریم که کد آن نیز به شرح زیر است:

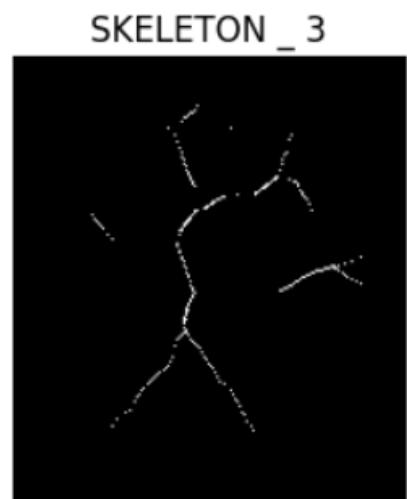
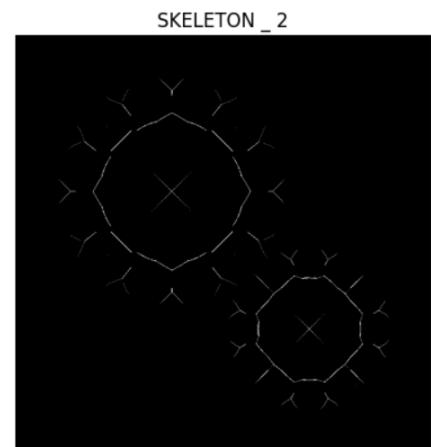
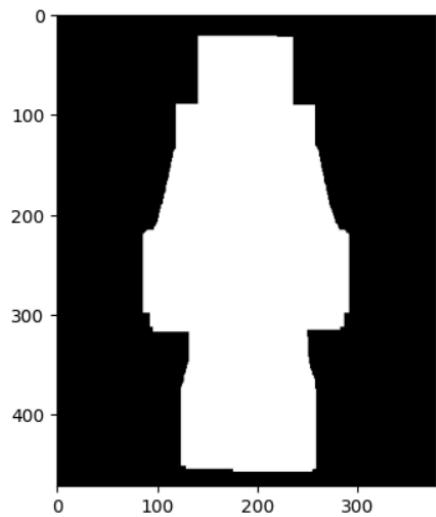
```

1 img1 = cv2.imread('./images/q8_5.jpg')
2 img2 = cv2.imread('./images/q8_6.png')
3 img3 = cv2.imread('./images/q8_7.png')
4 gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
5 gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
6 gray3 = cv2.cvtColor(img3, cv2.COLOR_BGR2GRAY)
7
8 _, binary_image_1 = cv2.threshold(gray1,100,255,cv2.THRESH_BINARY)
9 _, binary_image_2 = cv2.threshold(gray2,100,255,cv2.THRESH_BINARY)
10 _, binary_image_3 = cv2.threshold(gray3,100,255,cv2.THRESH_BINARY)
11
12 skeleton1, skeleton_list1 = skeletonize(binary_image_1)
13 backup1 = recreate(skeleton_list1, binary_image_1.shape)
14
15 skeleton2, skeleton_list2 = skeletonize(binary_image_2)
16 backup2 = recreate(skeleton_list2, binary_image_2.shape)
17
18 skeleton3, skeleton_list3 = skeletonize(binary_image_3)
19 backup3 = recreate(skeleton_list3, binary_image_3.shape)

```

که در تصویر بالا مقدار هر سه عکس را می خوانیم و به برای آن threshold می گذاریم و سپس برای هر کدام از عکس ها خروجی زیر دریافت می شود و اسکلت ها به صورت زیر خواهند بود.





```

1 def generate_binary_image(skeleton_list, kernel):
2     binary_image = np.zeros_like(skeleton_list[0], dtype=np.uint8)
3     for skeleton in skeleton_list:
4         binary_image = cv2.dilate(binary_image, kernel)
5         binary_image = cv2.bitwise_or(binary_image, skeleton)
6
7     return binary_image
8

1 kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
2 binary_image_reconstructed = generate_binary_image(skeleton_list2, kernel)
3 plt.imshow(binary_image_reconstructed, cmap='gray')

<matplotlib.image.AxesImage at 0x7fc77edeead0>


```

همانطور که در برخی از تصاویر بالا دیده می شود، برای بازسازی هر کدام از تصاویر نیز مانند فرمول گفته شده در اسلایدها باید اجتماع Sk ها را با گسترش kb به دست آوریم که کد آن نیز به صورت زیر می باشد:

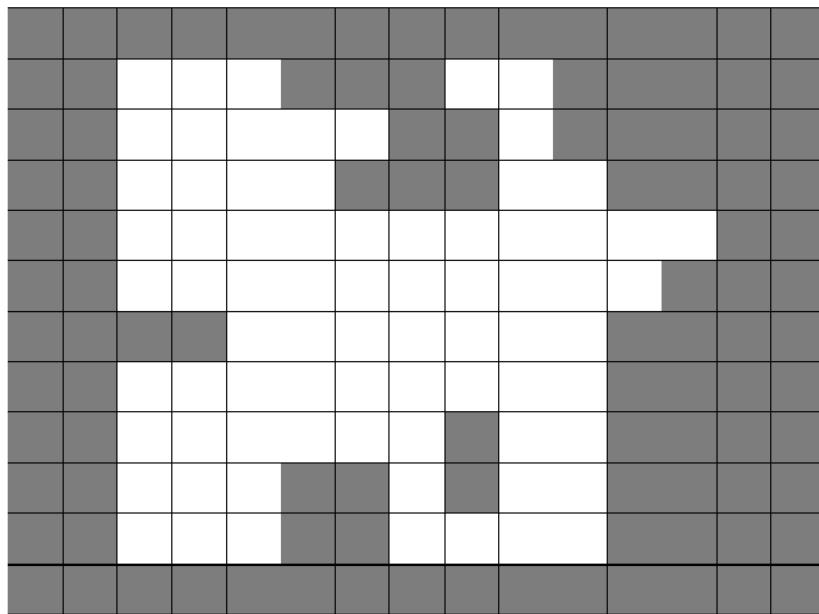
```

1 import numpy as np
2 import cv2
3
4 def recreate(intermediate_steps, output_shape):
5     reconstructed_image = np.zeros(shape=output_shape, dtype=np.uint8)
6     structuring_element = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
7     for step in intermediate_steps:
8         reconstructed_image = cv2.bitwise_or(step, reconstructed_image)
9         for _ in range(len(intermediate_steps)):
10             reconstructed_image = cv2.bitwise_or(reconstructed_image, cv2.dilate(step, structuring_element))
11
12     return reconstructed_image

```

(9)

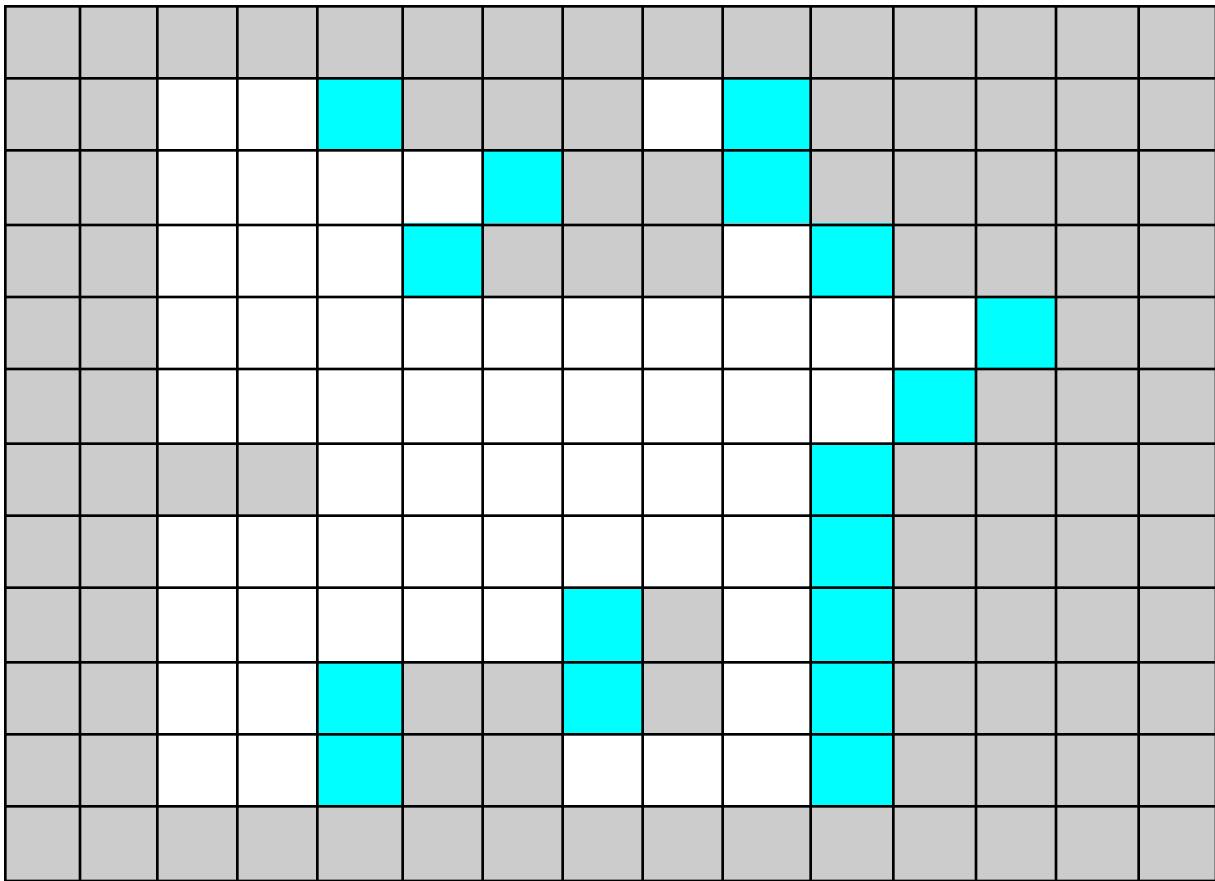
۹. با استفاده از عملگر hit or miss مرازهای تصویر زیر را به دست آورده و عملگرهای ساختاری مناسب به دست بیاورید. (سوال تئوری- ۵ نمره)



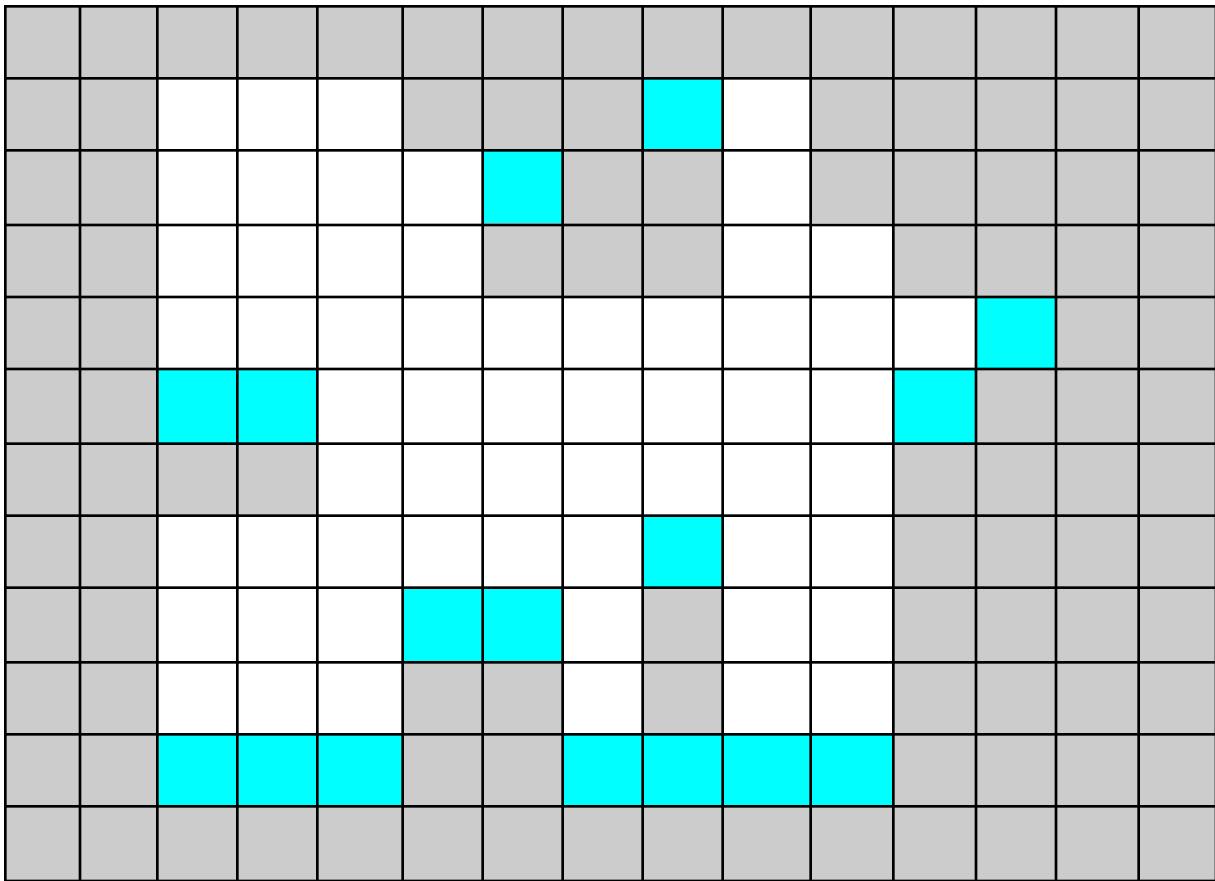
پاسخ:

برای این کار در ابتدا ۴ فیلتر $2 * 2$ در نظر می‌گیریم که هر کدام مرز مخصوص به خود را تشخیص می‌دهند. از آنجا که حاشیه‌ی تصویر همگی صفر هستند لذا نیازی به padding‌های مختلف نیست برای هر فیلتر نیز بدین صورت عمل می‌کنیم که پیکسل اصلی را همان نقطه‌ی مساوی با یک در نظر می‌گیریم و سپس باید نقطه‌ای که روی آن قرار می‌گیریم باید یک و نقطه‌ی منفی یک باید صفر و نقاط صفر نیز هر چیزی می‌توانند باشند. که با توجه به توضیحات گفته شده نقاط فیروزه‌ای برای هر کدام از فیلترها جواب می‌باشند.

1	1-
0	0

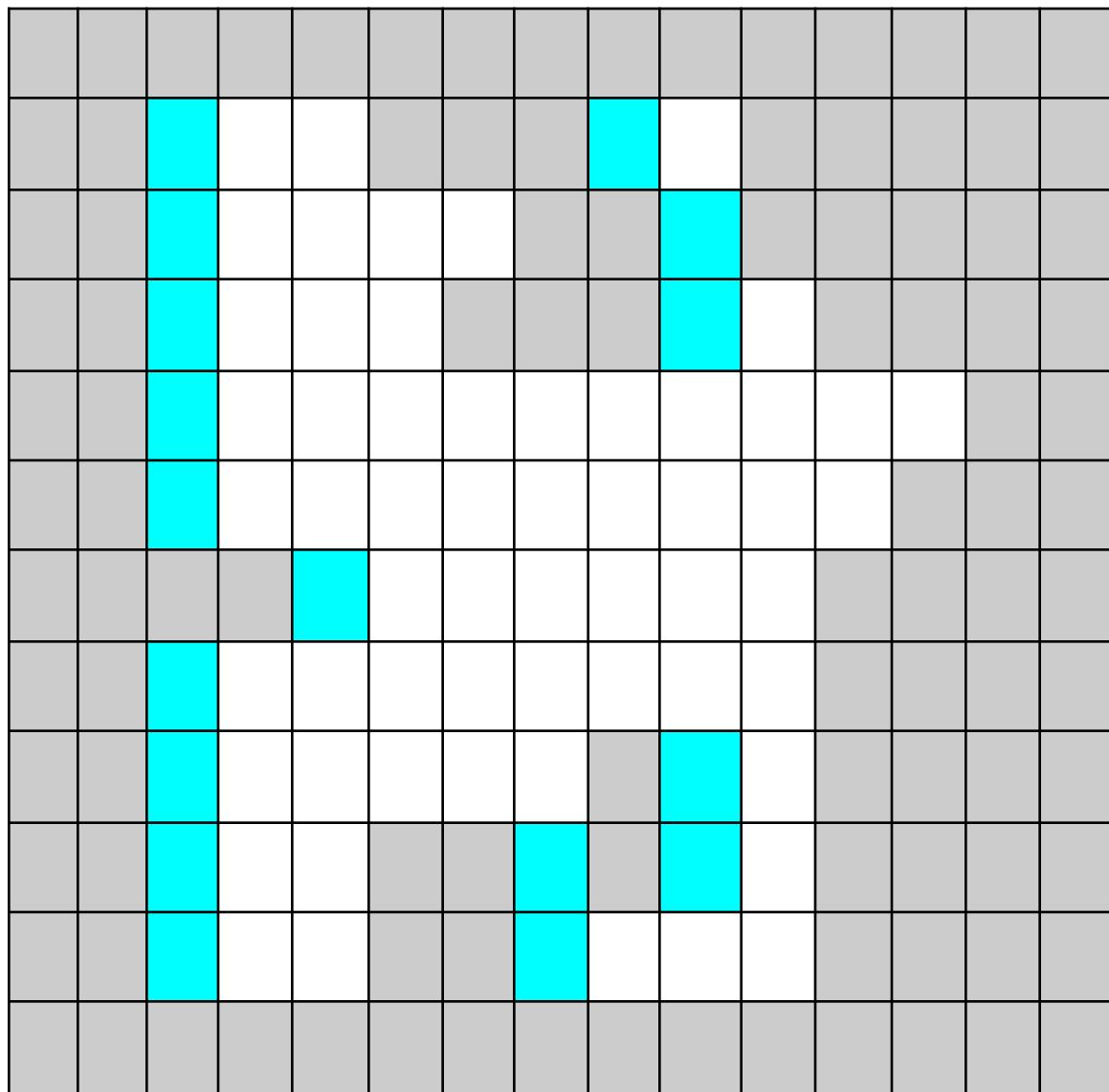


1	0
1-	0

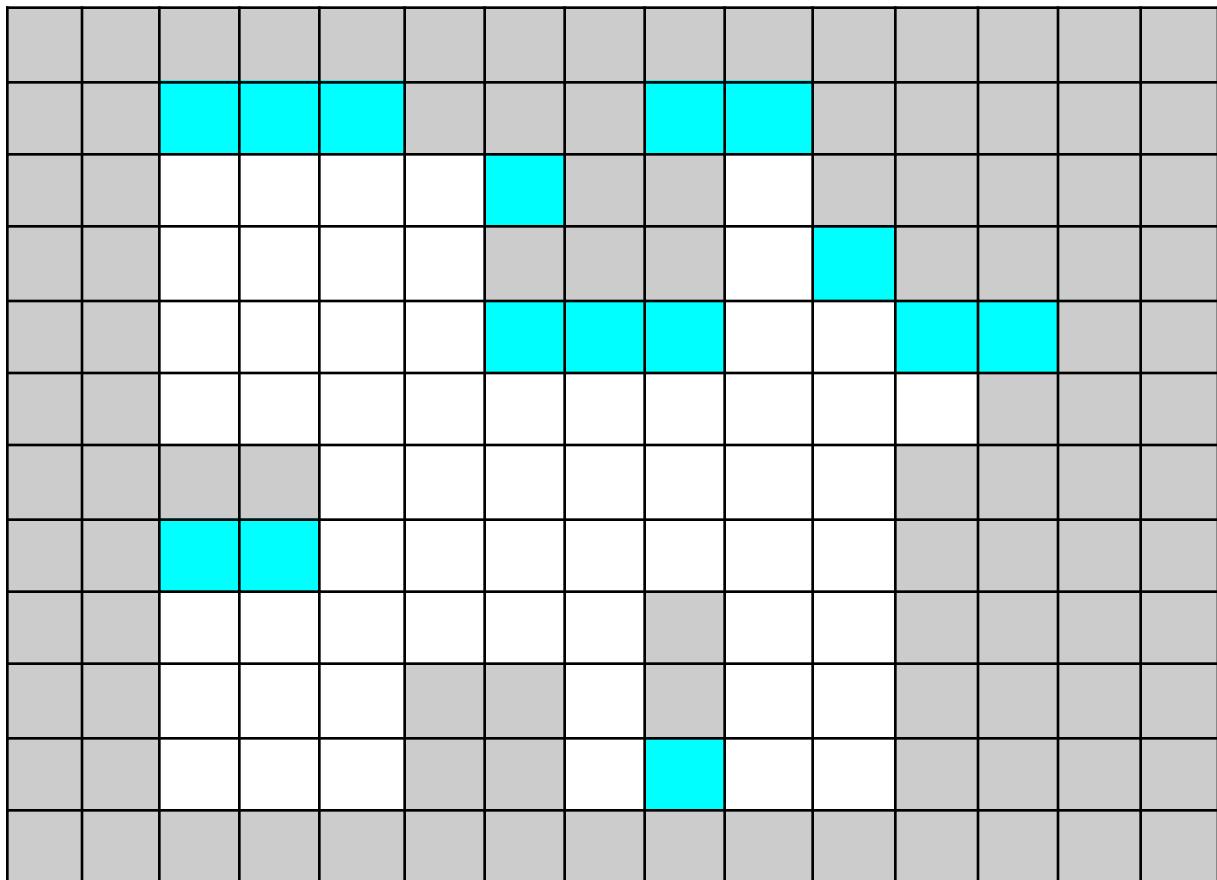


1-	1
----	---

0	0
---	---

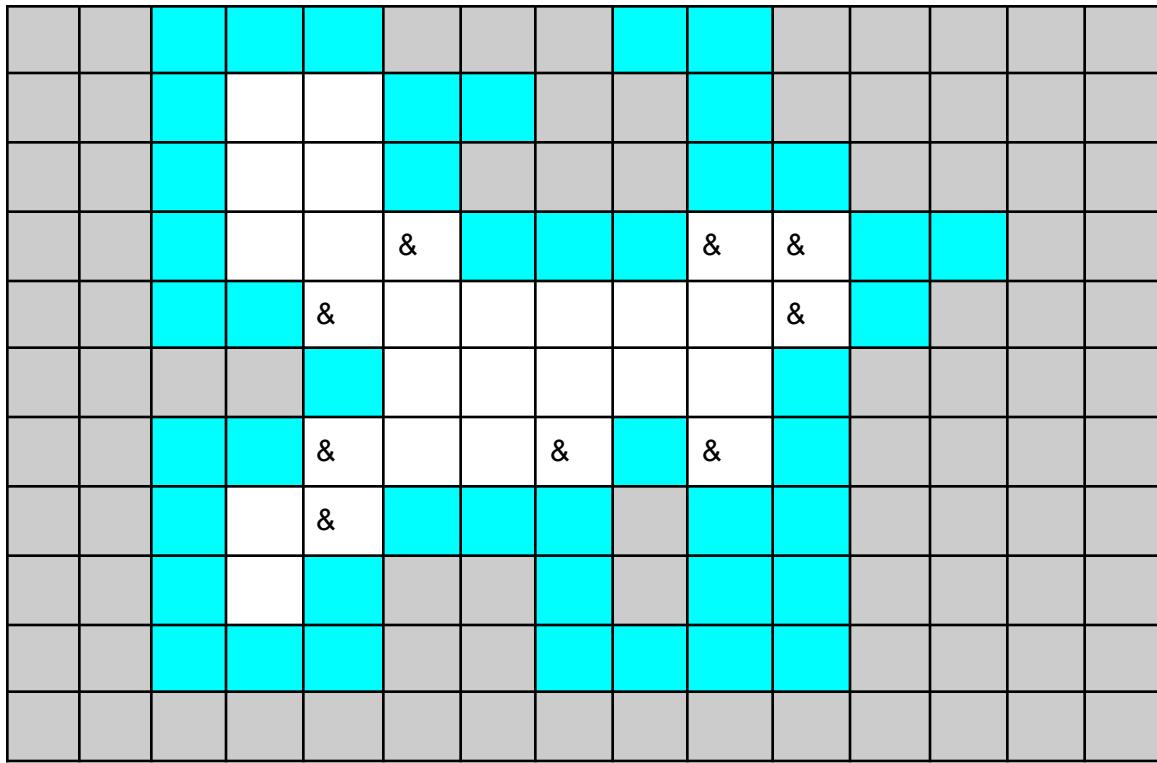


1-	0
1	0



حال از 4 شکل بالا اشکال فیروزه ای رنگ را اجتماع می گیریم و داریم:





که نقاط به دست آمده همان جواب ما می باشند و نقاط مرزی محسوب می شوند. به غیر از راه نوشته شده نیز می توانستیم از فیلتر 3 در 3 استفاده کنیم و یا نقاط گوشی ای را نیز حساب کنیم که برای این کار حتما باید از فیلتر 3 در 3 استفاده کنیم و به جواب های ما نقاطی که با علامت & نمایش داده شده اند نیز به جواب های ما افزوده می شوند.