به نام خدا

گزارش تمرین سری چهار درس بینایی کامپیوتر
نام مدرس: دکتر محمدرضا محمدی

فرزان رحمانی ۹۹۵۲۱۲۷۱

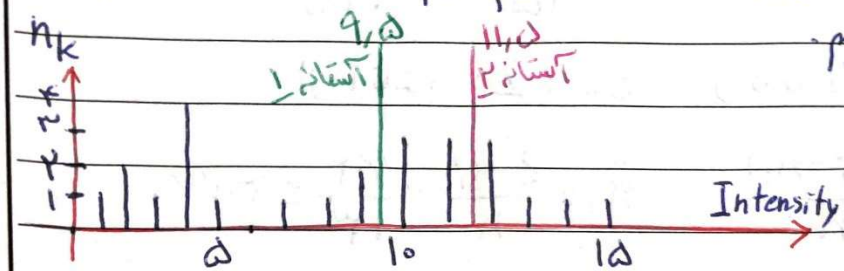
سوال ۱

(۱) الف)

مد

k	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
n_k	۰	۱	۲	۱	۴	۱	۰	۱	۱	۲	۳	۳	۳	۱	۱	۱

می‌توانیم n_k را بر $n=۲۵$ تقسیم کنیم تا normalize شود ولی ما اینجا نکردیم.



سطوح ۴ بیتی ↓ $L=۱۶=۲^۴$

$k\in[۰,۱۵]$

$$\bar{x}=avg=\frac{\sum k}{n}=\frac{۱+۲\times۲+۳+۴\times۴+۵+۷+۸+۹\times۲+۱۰\times۳+۱۱\times۳+۱۲\times۳}{۲۵}$$

$$+۱۳\times۱+۱۴\times۱+۱۵\times۱=۸,۱۲$$

$\bar{x}=avg=۸,۱۲$ میانگین

مد = ۴ چون دارای بیشترین تکرار است.

میانه = ۹ باید داده‌ها را مرتب کنیم و داده‌ی وسط (۱۳ام) را بیابیم

۱,۲,۲,۳,۴,۴,۴,۵,۷,۸,۹,**۹**,۱۰,۱۰,۱۰,۱۱,۱۱,۱۱,۱۲,۱۲,۱۲,۱۳,۱۴,۱۵

$$\sigma^۲=\frac{\sum(x_i-\bar{x})^۲}{n}$$

واریانس = ۱۶,۲۴۵۶

$$\sigma^۲=\frac{(۷,۱۲)^۲+۲(۹,۱۲)^۲+(۵,۱۲)^۲+۴(۴,۱۲)^۲+(۳,۱۲)^۲+(۱,۱۲)^۲+(۰,۱۲)^۲+۲(۰,۱۸)^۲}{۲۵}$$

$$+\frac{۳(۱,۸۸)^۲+۳(۲,۸۸)^۲+۳(۳,۸۸)^۲+(۴,۸۸)^۲+۵,۸۸^۲+۶,۸۸^۲}{۲۵}=\frac{۴۱۸,۶۴}{۲۵}$$

DATE: / / SUBJECT: NEGIN THE BEST QUALITY PAPER

$\sigma^2 = 14,7625$

threshold $= 9,5 \rightarrow w_1 = \frac{13}{25}$, $w_2 = \frac{12}{25}$ (ب

class 1: از ۵ تا ۹ class 2: از ۱۰ تا ۱۵

$\sigma_1^2 = \frac{\sum(x_i - \bar{x}_1)}{n}$ $\bar{x}_1 = \frac{\sum x_i}{n} = \frac{42}{13} \simeq 4,77$

$\sigma_1^2 = \frac{3,77^2 + 2(2,77)^2 + (1,77)^2 + (0,77)^2 + (0,23)^2 + (1,23)^2 + 2(2,23)^2}{13}$

$\sigma_1^2 = \frac{18,3013}{13} = 4,4291$

$\bar{x}_2 = \frac{\sum x_i}{n} = \frac{141}{12} = 11,75$ $\sigma_2^2 = \frac{\sum(x_i - \bar{x}_2)}{n}$

$\sigma_2^2 = \frac{3(1,75)^2 + 3(0,75)^2 + 3(0,25)^2 + (1,25)^2 + (2,25)^2 + (3,25)^2}{12}$

$\sigma_2^2 = \frac{28,2504}{12} = 2,35422$

ولی $\sigma_W^2 = w_1\sigma_1^2 + w_2\sigma_2^2$

$\sigma_{W_{9,5}}^2 = \frac{13}{25}(4,4291)^2 + \frac{12}{25}(2,35422)^2 = 4,05123481$

$\sigma_{W_{9,5}}^2 = 4,05123481$

NEGIN

آستانه ۹٫۵ بهتر

DATE: / / SUBJECT: NEGIN THE BEST QUALITY PAPER

threshold $\rightsquigarrow w_1 = \frac{19}{25}$, $w_2 = \frac{6}{25}$

class 1: 11 5 0 class 2: 15 6 12

$\bar{x}_1 = \frac{125}{19} = 8,\bar{3}$ $\sigma_1^2 = \frac{(7,\bar{3})^2 + 2(9,\bar{3})^2 + 3(5,\bar{3})^2 + 4(4,\bar{3})^2 +}{}$

$\frac{(5,\bar{3})^2 + (0,\bar{3})^2 + (0,\bar{6})^2 + 3(1,\bar{6})^2 + 3(2,\bar{6})^2}{19} = \frac{222,4\bar{3}150}{19}$

$\sigma_1^2 = 11,7176\bar{5}$ $\bar{x}_2 = \frac{78}{6} = 13$

$\sigma_2^2 = \frac{3\times1^2 + 0^2 + 1^2 + 2^2}{6} = \frac{78}{6} = 1,\bar{3}\bar{4}\bar{3}\bar{4}$

$\sigma_2^2 = \frac{8}{6} = 1,\bar{3}\bar{3}\bar{4}\bar{3}$

$\sigma_{w_{11,5}}^2 = \frac{19}{25}(11,7176\bar{5})^2 + \frac{6}{25}(1,\bar{3}\bar{4}\bar{3}\bar{3})^2 = 9,22\bar{5}3$

$\sigma_{w_{11,5}} = 9,22\bar{5}3$ آستانه بهتر

$9,2 \simeq \sigma_{w_{11,5}}^2 > \sigma_{w_{9,5}}^2 \simeq 4,6$ → سطح 9,5 آستانه بهتری است چرا که واریانس آن کمتر است. (واریانس بین پیکسل های هر کلاس کمتر است.)

NEGIN

سوال ۲

الف)

Sources:

https://www.researchgate.net/publication/277076039_Image_Binarization_using_Otsu_Thresholding_Algorithm
https://en.wikipedia.org/wiki/Otsu%27s_method
https://chat.openai.com/

از نظر سرعت Gaussian Otsu بسیار سریع تر از روش Otsu است. دلیل آن هم با توجه فرمولی که در صورت سوال آمده است قابل توجیح است.
(This approach is far faster than the optimal Otsus method.)

$$\begin{aligned}\sigma_B^2(t) &= \sigma^2 - \sigma_w^2(t) \\ &= w_b(t) * (\mu_b(t) - \mu))^2 + w_f(t) * (\mu_f(t) - \mu)^2 \\ &= w_b(t) * w_f(t) * (\mu_b(t) - \mu_f(t))^2\end{aligned}$$

از نظر دقت نیز با توجه به نتایج آزمایش لینک مقاله بالا Gaussian Otsu دقت بیشتری نسبت به Otsu دارد به خصوص در تصاویر که دارای two modes in their histogram هستند.

مقایسه مفصل تر:

The results show that Otsu's method correctly classified the pixels into foreground and background regions for 68% of images while Gaussian Otsu's method 100% accurately detected foreground regions.
Table 1 shows the thresholding values for each thresholding approach, and Table 2 shows the typical results after thresholding operations using both methods.
By observing histograms and the thresholding values in Table 1, it is clear that the Guassian Otsu' method works well in images that have bimodal distributions (have two modes in their histograms) while Otsu's method gives incorrect threshold value that fails the binarization process . From the result binary images in Table 2, it is also clear that Gaussian Otsu's method is able to generate a binary image for this cases while Otsu's method falsely classifies the background pixles as foreground (object). Image 'womandarkhair' with two modes in its histogram, is an example of Otsu's method dificulty in detecting the background and the object

correctly. The other falsify classification with Otsu’s method occurs in "walkbridge" image where even visuallly it is hard to determine the parts of the image belonging to object and back-ground. The results in Table 2 also show that both methods perform same for images with more complex histogram patterns (multiple modals). ‘Lena’ and ’womanblonde’ are good examples for multiple modals.
Otsu’s method attempts to find the best threshold value by minimizing within class vaiance while Gaussian Otsu tries to find the best thresholding value by maximizing between class variance.

Gaussian Otsu’s method significantly performs better on the images with two modes in their histogram and otherwise both methods find the same threshold value and perform same in the classification of images with multiple modals.

توضیح مختصر درباره دو الگوریتم:

Otsu's method

Otsu's thresholding is a commonly used image processing technique for automatic thresholding of grayscale images. It is named after its inventor, Nobuyuki Otsu, and is based on the idea of minimizing the intra-class variance of two classes of pixels, where one class represents the background and the other represents the foreground. The threshold value that separates the two classes is chosen to minimize the sum of the intra-class variances.

In simpler terms, Otsu's thresholding finds the optimal threshold value that separates the background and foreground pixels in an image, such that the variance within each class is minimized and the variance between the two classes is maximized.

The Otsu's thresholding algorithm involves calculating the histogram of the input image, and then iterating through all possible threshold values to calculate the variance between the two classes of pixels. The threshold value that corresponds to the minimum intra-class variance is chosen as the optimal threshold.

Otsu's thresholding is widely used in various image processing applications such as image segmentation, object detection, and feature extraction. It is a simple and effective method for thresholding grayscale images, especially when the threshold value cannot be determined based on prior knowledge of the image.

Gaussian Otsu

Gaussian OTSU is a variant of the Otsu's thresholding method, which is used to automatically calculate an optimal threshold value for image segmentation.

The Gaussian OTSU algorithm works by calculating the histogram of the input image, then iterating over all possible threshold values and computing the between-class variance for each threshold. The threshold that maximizes the between-class variance is chosen as the optimal threshold value.

The between-class variance is defined as the weighted sum of variances between the two classes of pixels that are separated by the threshold.

Here is the pseudo-code for the Gaussian OTSU algorithm:

1. Convert the input image to grayscale.

2. Compute the histogram of the grayscale image.

3. Compute the cumulative distribution function (CDF) of the histogram.

4. Compute the mean and variance of the pixel intensities for each possible threshold value.

5. For each threshold value, compute the between-class variance using the formula:

   sigma^2 = w0 * w1 * (mu0 - mu1)^2

   where:

   - w0 and w1 are the probabilities of the two classes separated by the threshold

   - mu0 and mu1 are the means of the two classes separated by the threshold

6. Find the threshold value that maximizes the between-class variance.

7. Apply the threshold to the grayscale image to obtain a binary image.

The resulting binary image will have foreground (object) pixels with a value of 1 and background pixels with a value of 0.

The threshold obtained using Gaussian OTSU can be used to segment the input image into two classes: foreground and background. Pixels with intensities above the threshold are classified as foreground, while pixels with intensities below the threshold are classified as background.

ب)

بله، در الگوریتم Otsu، به حداقل رساندن واریانس درون کلاسی(minimizing within class variance) معادل به حداکثر رساندن واریانس بین کلاسی(maximizing between class variance) است. این به این دلیل است که واریانس کل تصویر را می توان به مجموع واریانس درون کلاسی و واریانس بین کلاسی تجزیه کرد. به حداقل رساندن واریانس درون کلاسی معادل به حداکثر رساندن واریانس بین طبقاتی است زیرا مجموع واریانس درون کلاسی و واریانس بین طبقاتی ثابت است.

واریانس درون کلاسی(within class variance) واریانس درون یک کلاس را نشان می‌دهد، به این معنی که همگنی پیکسل‌های درون هر کلاس را اندازه‌گیری می‌کند. به حداقل رساندن واریانس درون کلاسی منجر به یک کلاس همگن تر می شود. از سوی دیگر، واریانس بین کلاسی(between class variance) واریانس بین کلاس ها را نشان می دهد، به این معنی که تفکیک پذیری کلاس ها را اندازه می گیرد. به حداکثر رساندن واریانس بین طبقاتی(کلاسی) منجر به جدایی بهتر بین طبقات(کلاس ها) می شود.

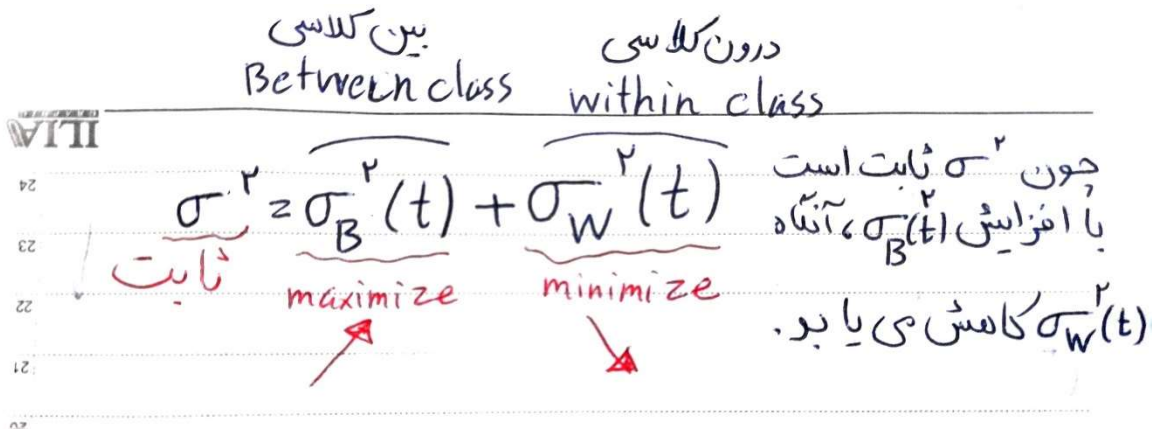
بنابراین، با به حداقل رساندن واریانس درون کلاسی، به طور غیرمستقیم واریانس بین طبقاتی را به حداکثر می‌رسانیم که منجر به انتخاب آستانه بهتر در الگوریتم Otsu می‌شود.

پس کمینه کردن واریانس درون کلاسی (within-class variance) در الگوریتم Otsu برابر است با بیشینه کردن واریانس بین کلاسی(between-class variance) .

برای توضیح این موضوع، در نظر بگیرید که در الگوریتمOtsu ، به دنبال یافتن آستانه‌ای هستیم که واریانس درون کلاسی برای آستانه برابر با آستانه، کمینه شود. در واقع، هدف ما این است که دو کلاس ایجاد شده توسط آستانه، به گونه‌ای تعریف شوند که واریانس داخل کلاس‌ها، حداقل شود.

اگر به جای کمینه کردن واریانس درون کلاسی، به دنبال بیشینه کردن واریانس بین کلاسی باشیم، به دنبال آستانه‌ای هستیم که بیشترین تفاوت بین دو کلاس ایجاد شده توسط آستانه، را ایجاد کند. با توجه به اینکه در الگوریتمOtsu ، تنها دو کلاس وجود دارند (پیکسل‌هایی که بیشتر یا کمتر از آستانه هستند)، بنابراین کمینه کردن واریانس درون کلاسی با بیشینه کردن واریانس بین کلاسی، معادل است.

به طور خلاصه، در الگوریتمOtsu ، هدف ما کمینه کردن واریانس داخل کلاس‌ها است، که در واقع بیشینه کردن تمایز دو کلاس ایجاد شده توسط آستانه را نشان می‌دهد.



سوال ۳

sources: https://en.wikipedia.org/wiki/Region_growing

معیار اختلاف برای رشد ناحیه

- می‌توان رنگ پیکسل مورد نظر را با رنگ پیکسل بذر مقایسه کرد و اگر اختلاف آنها از حدی کمتر بود به ناحیه اضافه شوند
- این روش معادل با این است که ابتدا تصویر را بر اساس اختلاف با رنگ مورد نظر باینری کرده و سپس ناحیه متصل به این پیکسل را استخراج کنیم

۲۷

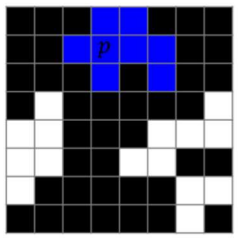
از روندی که در اسلاید بالا آمده است برای پیاده سازی استفاده کردم. همچنین برای یافتن عناصر متصل از الگوریتم زیر استفاده نمودم.

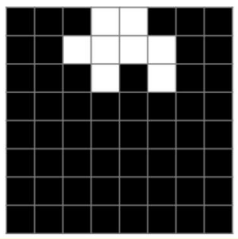
استخراج یک ناحیه متصل

```
// Finding the connected component containing an
   object pixel p
1. Initialize
    1. Create a result set S that contains only p
    2. Create a Visited flag at each pixel, and
       set it to be False except for p
    3. Initialize a queue (or stack) Q that
       contains only p.
2. Repeat until Q is empty:
    1. Pop a pixel x from Q.
    2. For each unvisited object pixel y
       connected to x, add y to S, set its flag
       to be visited, and push y to Q.
3. Output S
```

$S = \{(1,3), (0,3), (1,2), \dots\}$

$Q = \{\ \}$

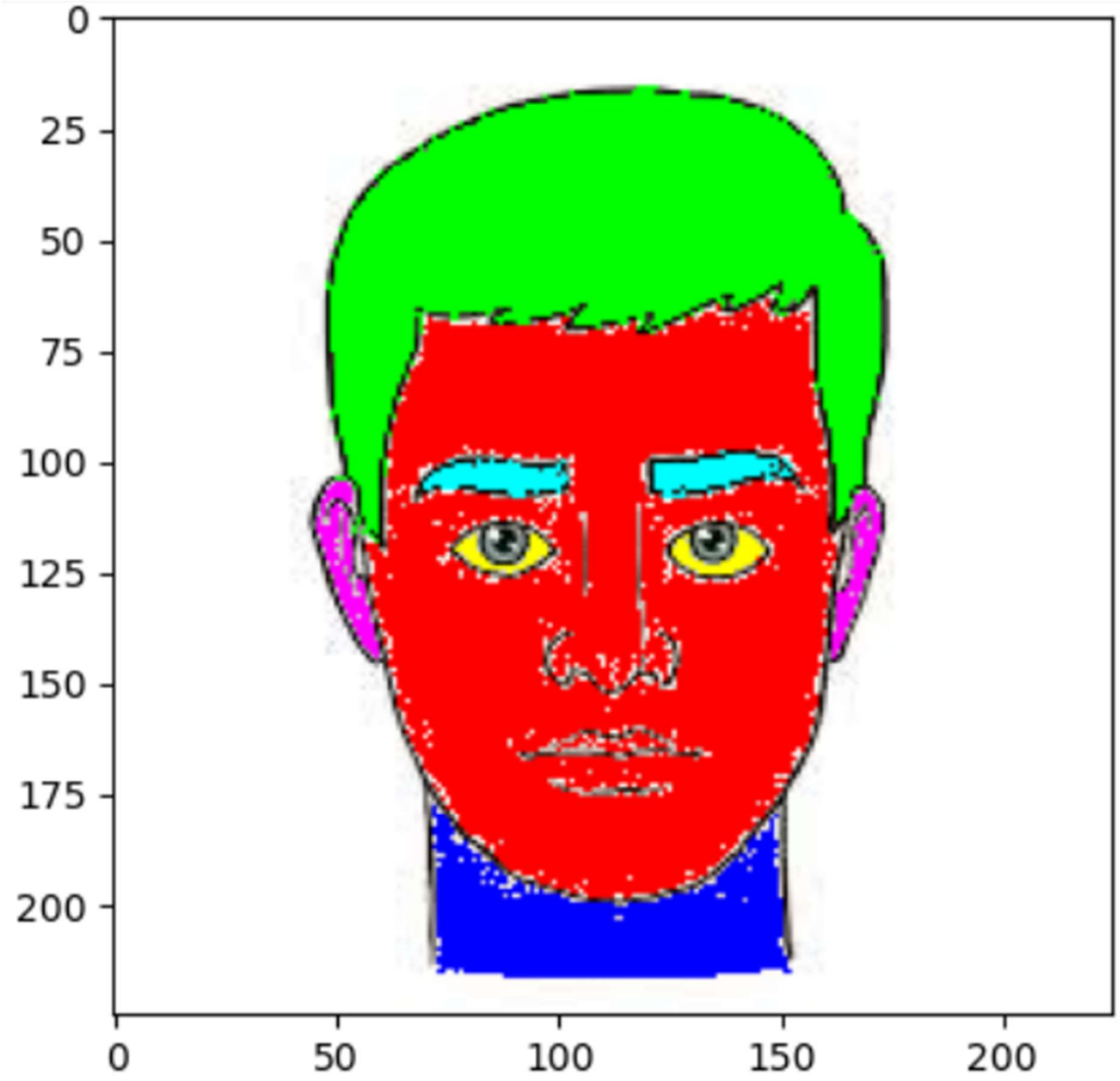




۱۹

ابتدا با استفاده از روش آزمون و خطا چندین seed به همراه آستانه های متناظر و رنگ های دلخواه انتخاب کردم. سپس الگوریتم بالا را برای هر sedd اجرا کردم. برای محاسبات ابتدا به floatتبدیل کردم. بعد تفاوت رنگ از دانه را پیدا کردم. سپس قدر مطلق تفاوت را محاسبه کردم. بعد از روی قدر مطلق تفاوت سه کانال برای هر پیکسل با میانگین گیری تفاوت خالص را پیدا کردم. بعد از آن برای یافتن اجزا متصل از الگوریتم bfs یا dfs میتوان استفاده کرد. فقط باید شرط های همسایگی و حدکثر تفاوت با آستانه را اعمال کنیم و همچنین می توان از همسایگی چهار تایی یا هشت تایی استفاده کرد که من از ۸ تایی بهره بردم.

نتیجه:



سوال ۴

الف)

برای حل سوال از فرمول های زیر استفاده می کنیم:

- عملگر گسترش (dilate) برای گسترش مجموعه A توسط B به صورت زیر تعریف می‌شود:

$$A \oplus B = \left\{ z \middle| (\hat{B})_z \cap A \neq \emptyset \right\}$$

anchor

- این رابطه به مفهوم بدست آوردن انعکاس B حول مرکز (لنگر) خودش و جابجایی آن به اندازه z است که اگر این نسخه از B دارای اشتراک با A بود، z جزء مجموعه جدید خواهد بود

- در یک تصویر سطح خاکستری، عملگر مورفولوژی گسترش به صورت زیر تعریف می‌شود

$$dst(x,y) = \max_{(x',y') \in SE} src(x+x', y+y')$$

- عملگر سایش (erode) برای فرسایش مجموعه A توسط B به صورت زیر تعریف می‌شود:

$$A \ominus B = \{ z | (B)_z \subseteq A \}$$

- بنابراین سایش مجموعه A توسط B شامل مجموعه نقاطی است که به ازای آنها B به طور کامل درون A قرار می‌گیرد

- عملگر سایش برای تصاویر سطح خاکستری

$$dst(x,y) = \min_{(x',y') \in SE} src(x+x', y+y')$$

تصویر اصلی پس از اعمال padding:

چون سایز کرنل ۳ در ۳ است پس از هر طرف ۱ واحد padding میزنیم.(3 // 2 = 1)

۶۰	۶۰	۷۰	۶۰	۶۰	۷۰	۶۰	۶۰	۶۰	۶۰
۶۰	۶۰	۷۰	۶۰	۶۰	۷۰	۶۰	۶۰	۶۰	۶۰
۶۰	۶۰	۷۰	۷۰	۷۰	۷۰	۷۰	۷۰	۶۰	۶۰
۶۰	۶۰	۷۰	۶۰	۷۰	۷۰	۷۰	۷۰	۷۰	۷۰
۸۰	۸۰	۶۰	۸۰	۶۰	۷۰	۸۰	۷۰	۷۰	۷۰
۶۰	۶۰	۷۰	۷۰	۶۰	۷۰	۶۰	۶۰	۶۰	۶۰
۶۰	۶۰	۷۰	۸۰	۶۰	۸۰	۷۰	۶۰	۶۰	۶۰
۷۰	۷۰	۶۰	۸۰	۶۰	۶۰	۸۰	۶۰	۶۰	۶۰
۶۰	۶۰	۷۰	۷۰	۸۰	۶۰	۸۰	۶۰	۷۰	۷۰
۶۰	۶۰	۷۰	۷۰	۸۰	۶۰	۸۰	۶۰	۷۰	۷۰

تصویر حاصل پس از افزایش:

۷۰	۷۰	۷۰	۷۰	۷۰	۷۰	۷۰	۷۰
۷۰	۷۰	۷۰	۷۰	۷۰	۷۰	۷۰	۷۰
۸۰	۸۰	۸۰	۸۰	۸۰	۸۰	۸۰	۷۰
۷۰	۸۰	۷۰	۷۰	۸۰	۷۰	۷۰	۷۰
۷۰	۸۰	۸۰	۸۰	۸۰	۸۰	۷۰	۶۰
۷۰	۸۰	۸۰	۸۰	۸۰	۸۰	۸۰	۶۰
۷۰	۸۰	۸۰	۸۰	۸۰	۸۰	۸۰	۷۰
۷۰	۷۰	۸۰	۸۰	۸۰	۸۰	۸۰	۷۰

تصویر حاصل پس از سایش:

۶۰	۶۰	۶۰	۶۰	۶۰	۶۰	۶۰	۶۰
۶۰	۶۰	۶۰	۶۰	۶۰	۶۰	۶۰	۶۰
۶۰	۶۰	۷۰	۶۰	۷۰	۷۰	۶۰	۶۰
۶۰	۶۰	۶۰	۶۰	۶۰	۷۰	۷۰	۷۰
۶۰	۶۰	۶۰	۶۰	۶۰	۷۰	۶۰	۶۰
۶۰	۶۰	۶۰	۶۰	۶۰	۶۰	۶۰	۶۰
۶۰	۶۰	۶۰	۶۰	۶۰	۶۰	۶۰	۶۰
۶۰	۶۰	۶۰	۶۰	۶۰	۶۰	۶۰	۶۰

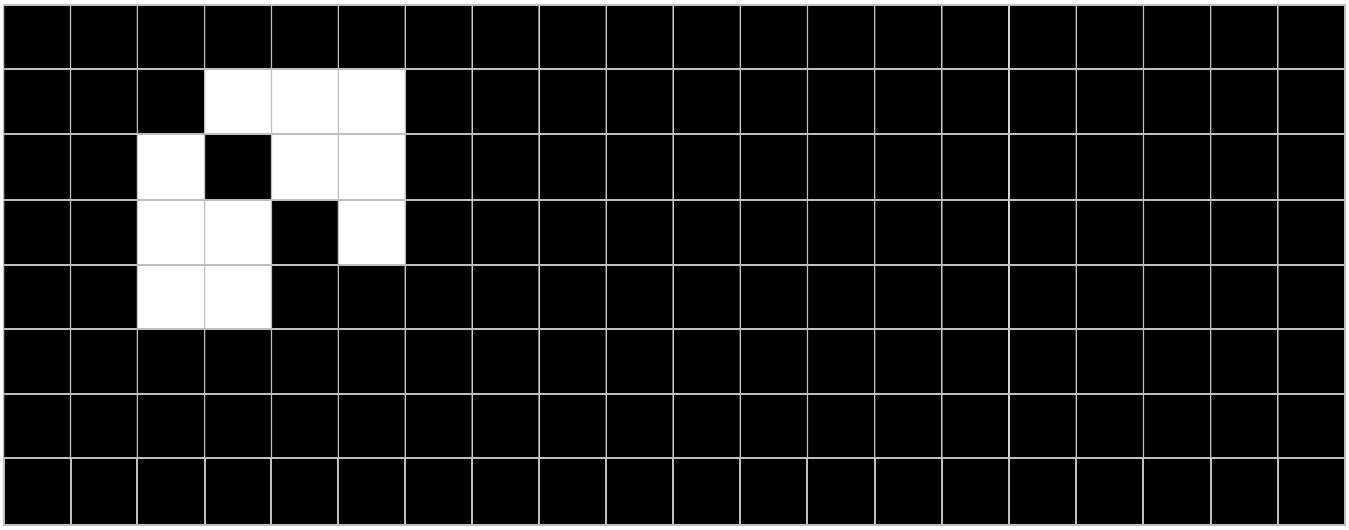
ب) در تمام قسمت های این سوال فرض بر zero padding در نظر گرفته شده است.

- عملگر باز (opening) برای حذف جزئیات کوچک و هموار کردن محیط نواحی تعریف شده است

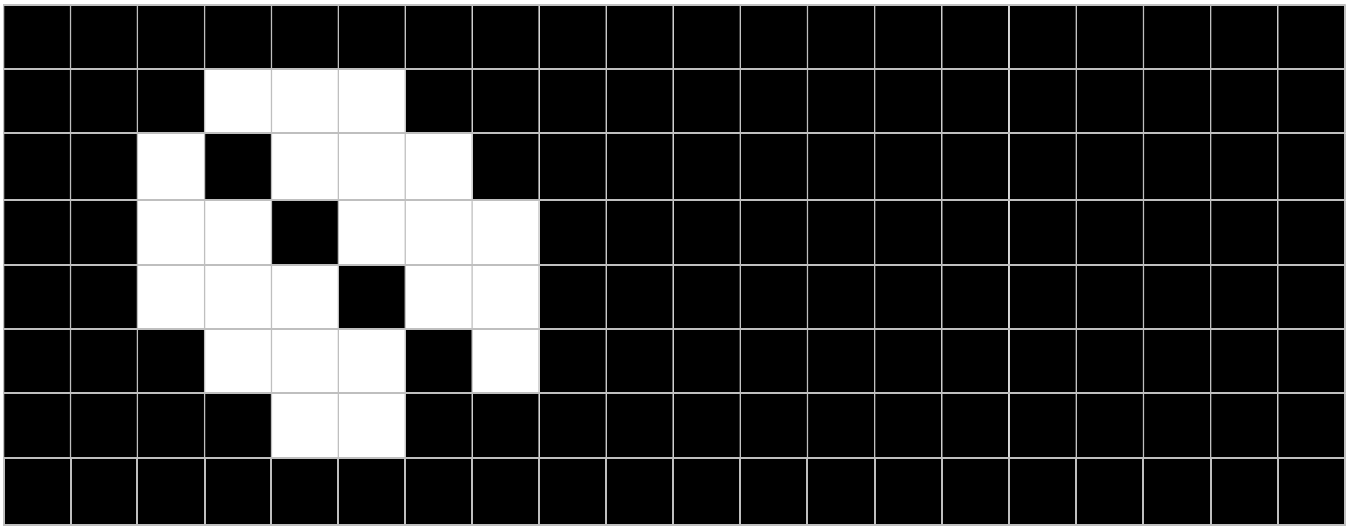
$$A \circ B = (A \ominus B) \oplus B$$

- این عملگر ناحیه‌های سفید که در احاطه پیکسل‌های سیاه هستند را حذف می‌کند

ابتدا تصویر را سایش می دهیم:



سپس تصویر حاصل را گسترش می دهیم (ابتدا کرنل را ۱۸۰ درجه حول لنگر می چرخانیم) تا حاصل عملگر باز به دست آید:



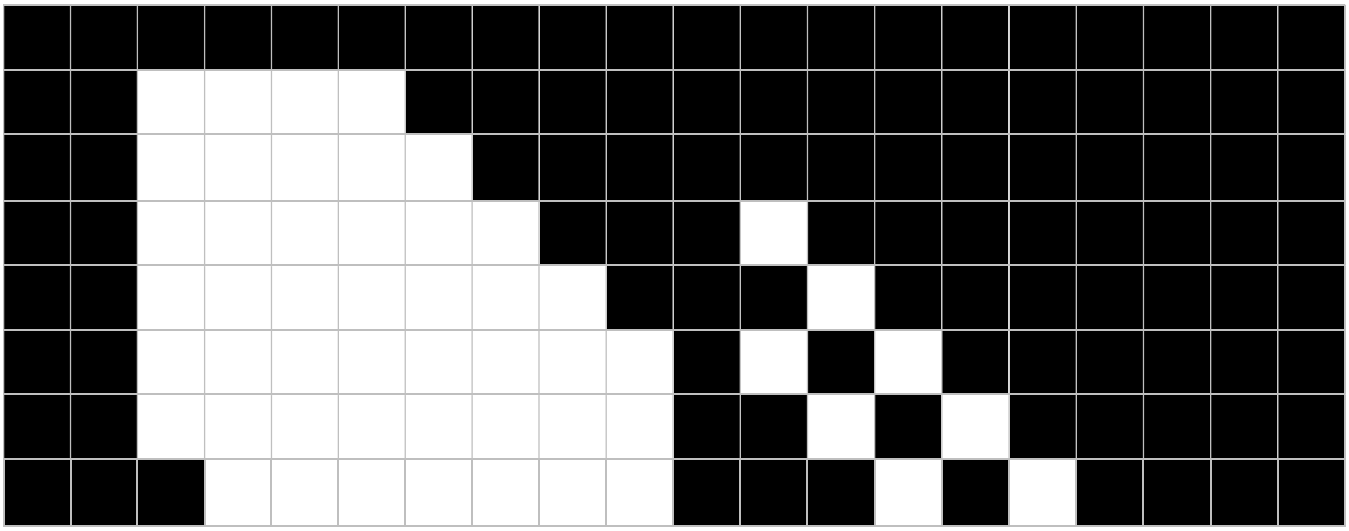
عملگر بسته

- عملگر بسته (closing) برای حذف حفره‌های کوچک و هموار کردن محیط نواحی تعریف شده است

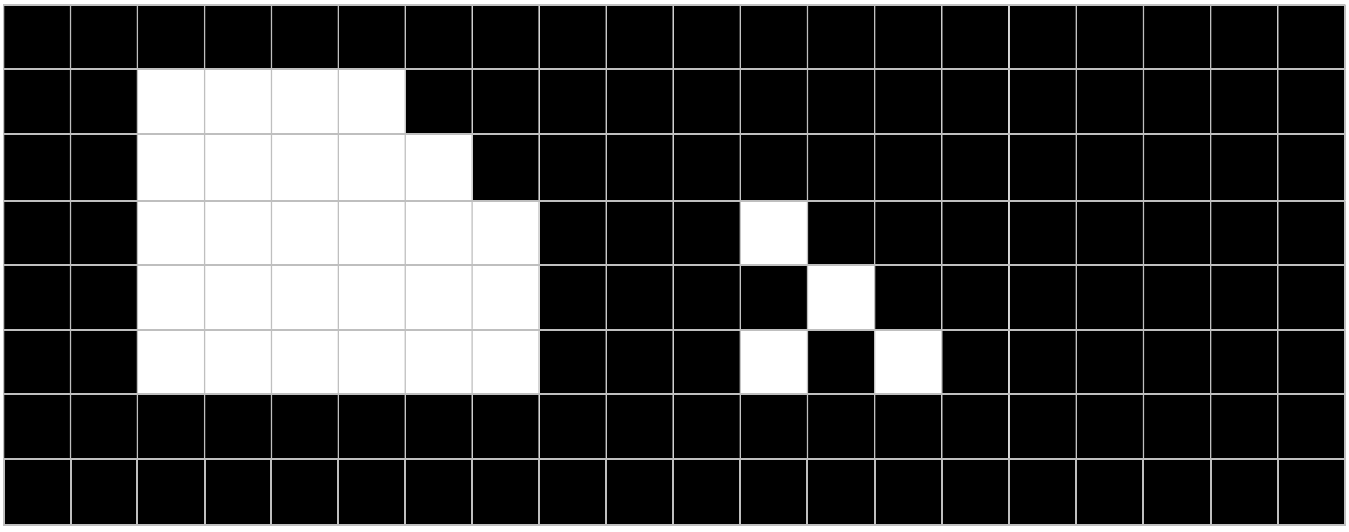
$$A \cdot B = (A \oplus B) \ominus B$$

- این عملگر ناحیه‌های سیاه که در احاطه پیکسل‌های سفید هستند را حذف می‌کند

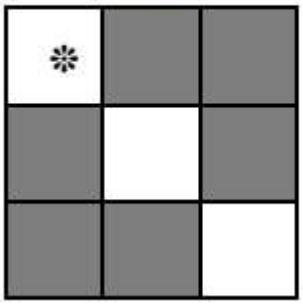
ابتدا تصویر را گسترش(ابتدا کرنل را ۱۸۰ درجه حول لنگر می چرخانیم) می دهیم:

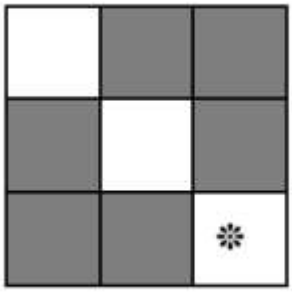


سپس تصویر حاصل را سایش می دهیم تا حاصل عملگر باز به دست آید:



SE





rotated SE

سوال ۵

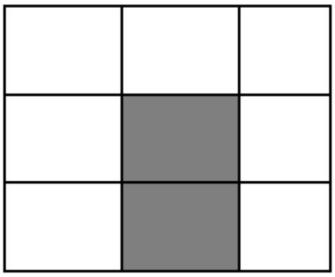
الف) در این سوال فرض میکنیم پیکسل های سفید مقدار پایینی (0) دارند و پیکسل های سیاه مقدار بالایی (1) دارند.

چون می خواهیم بدون تغییر پیکسل های اصلی شامل ۰ ، پیکسل های فرعی وسط را از بین ببریم باید از عملگر باز استفاده کنیم.

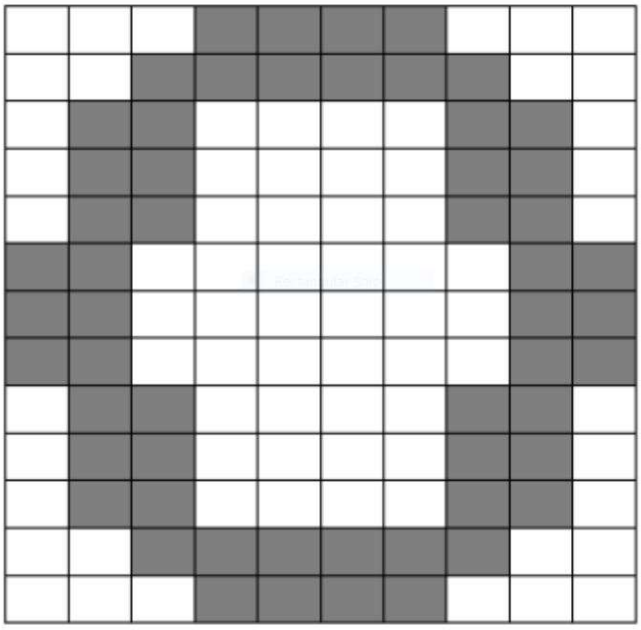
- عملگر باز (opening) برای حذف جزئیات کوچک و هموار کردن محیط نواحی تعریف شده است

$$A \circ B = (A \ominus B) \oplus B$$

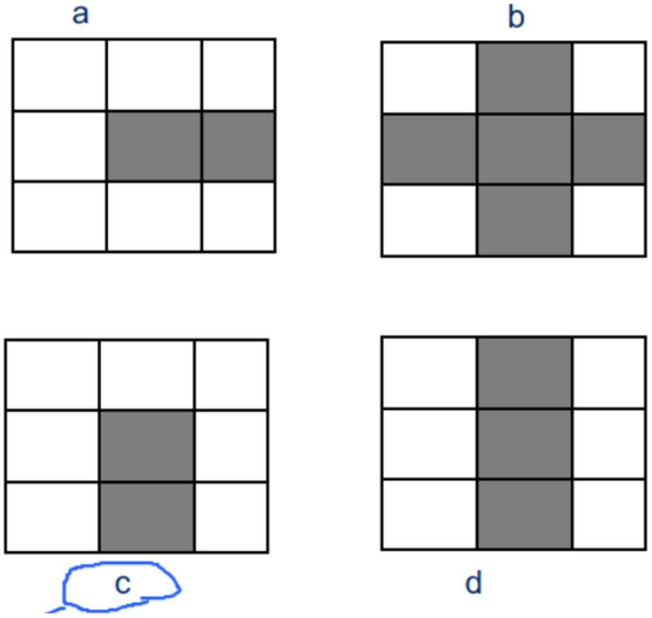
عنصر ساختاری باید به گونه ای باشد که خط وسط حذف شود که با بررسی عناصر ساختاری پی میبریم که باید از عنصر ساختاری زیر باید استفاده کنیم.



عملگر باز(opening) را روی تصویر با استفاده از عنصر ساختاری بالا اعمال می کنیم و نتیجه زیر حاصل می شود.



چهار عنصر ساختاری به صورت‌های زیر وجود دارد.



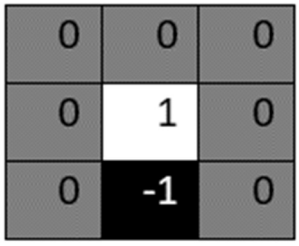
توضیحات بیشتر:

برای به دست آوردن رقم ۰ از تصویر باینری داده شده با اعمال عملگر باز مورفولوژیکی با یک عنصر ساختاری باینری، باید عنصر ساختاری را انتخاب کنیم که بتواند قسمت های نازک وسط را حذف کند و شکل را تا حد امکان به رقم ۰ نزدیک کند.
در بین عناصر ساختاری داده شده، گزینه (c) مناسب ترین گزینه است.

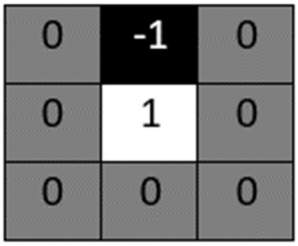
- گزینه a میتواند خط عمودی را حذف کند ولی در این مثال کاربردی ندارد و تفاوتی ایجاد نمیکند.
- گزینه b خط اضافی را حذف میکند ولی همزمان قسمت هایی از خود صفر را هم حذف میکند.
- گزینه c خط اضافه را حذف می کند ولی خود صفر کامل باقی می ماند.
- گزینه d نیز میتواند خط اضافی را حذف کند ولی همزمان قسمت هایی از خود صفر را هم حذف میکند.

ب)

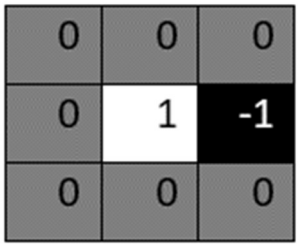
عنصر ساختاری برای یافتن لبه پایین:



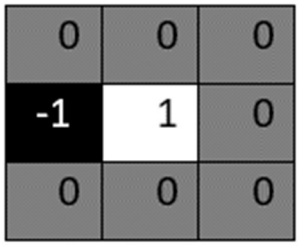
عنصر ساختاری برای یافتن لبه بالا:



عنصر ساختاری برای یافتن لبه راست:



عنصر ساختاری برای یافتن لبه چپ:



از عنصر های ساختاری بالا و عملگر hit-or-miss برای یافتن لبه های راست، چپ، بالا و پایین استفاده میکنیم و سپس اجتماع چهار تصویر حاصل را می گیریم که حاصل لبه های تصویر را بدست می آورد. (1:hit, 0:don’t care, -1:miss)

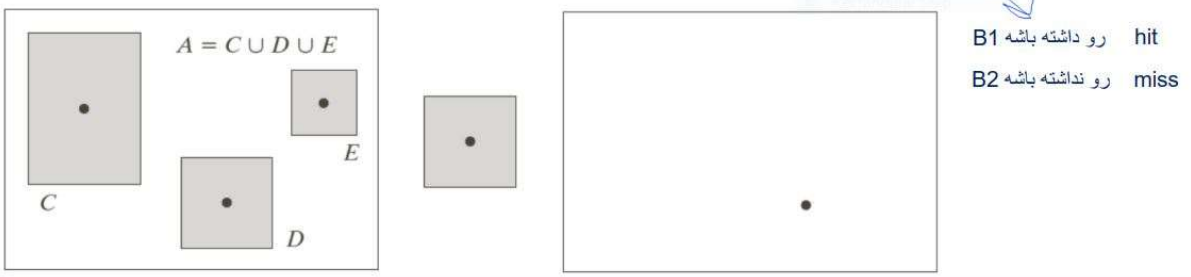
فرمول عملگر hit-or-miss:

عملگر Hit-or-Miss

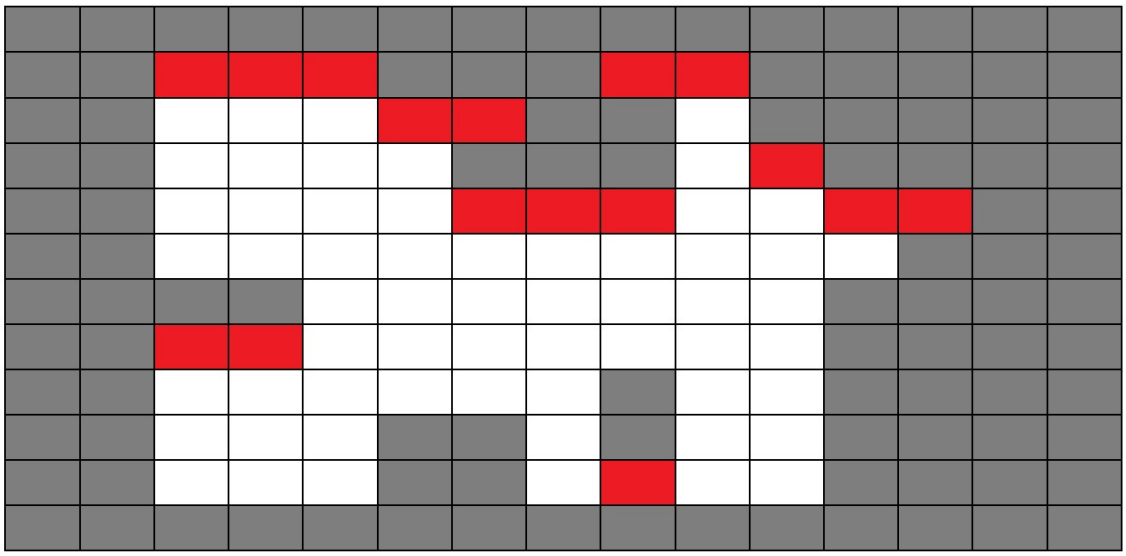
- عملگر Hit-or-Miss یک پردازش مورفولوژی برای تشخیص شکل یک ناحیه است و از آن برای استخراج الگویی در تصویر استفاده می‌شود

$$(A \circledast B) = (A \ominus X) \cap (A^c \ominus (W - X))$$

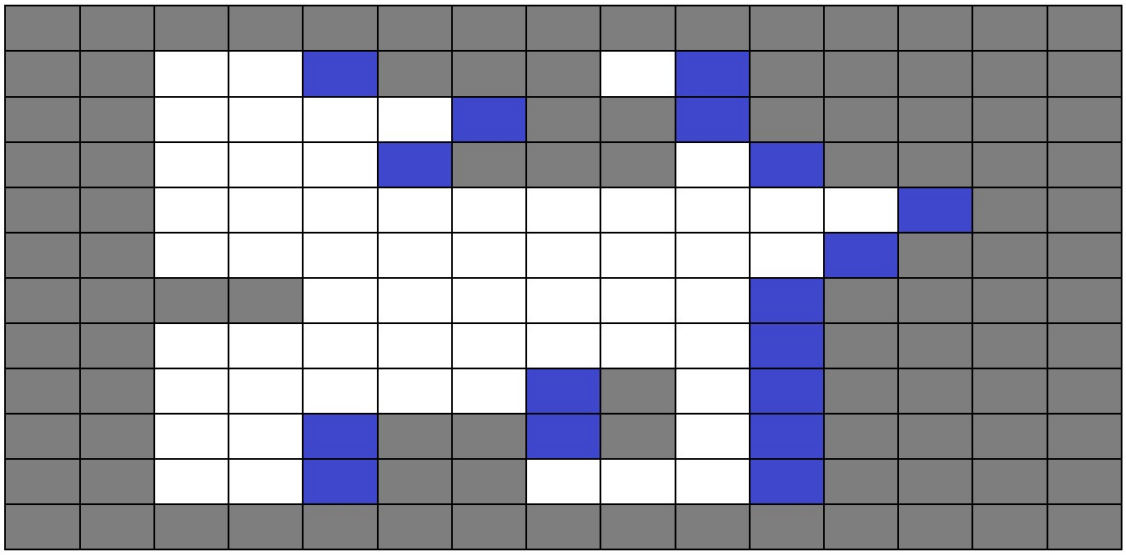
$$(A \circledast B) = (A \ominus B_1) \cap (A^c \ominus B_2)$$



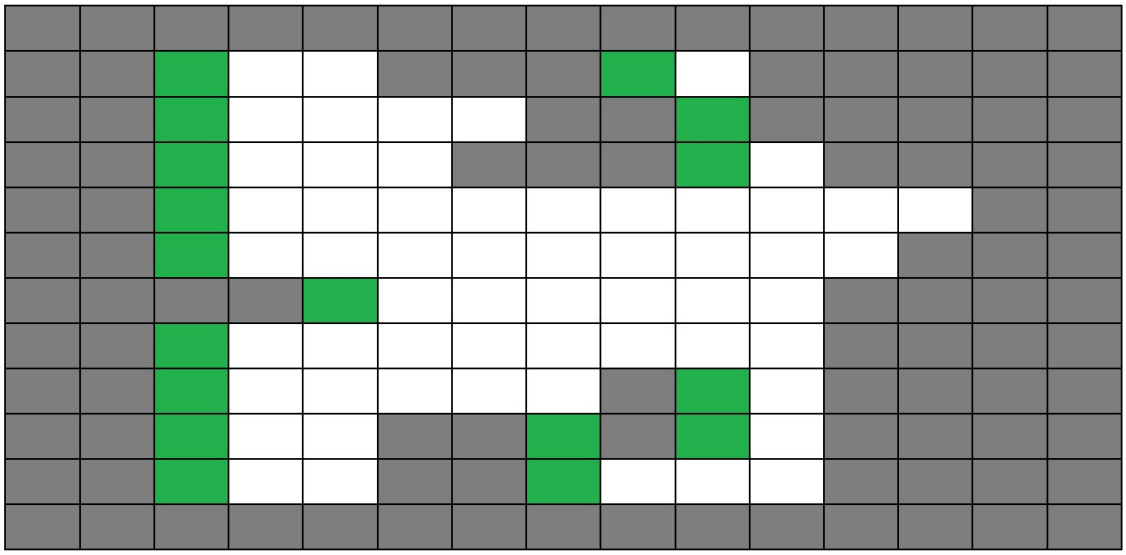
حال عملگذ hit-or-miss را با عناصر ساختاری بالا انجام می دهیم و نتایج زیر حاصل میشوند.
لبه بالا:



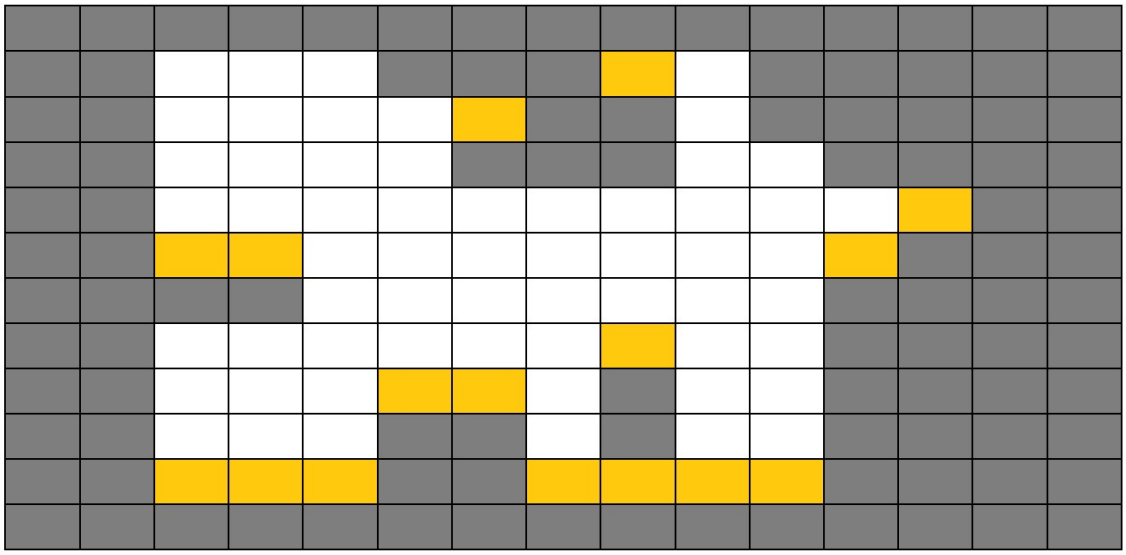
لبه راست:



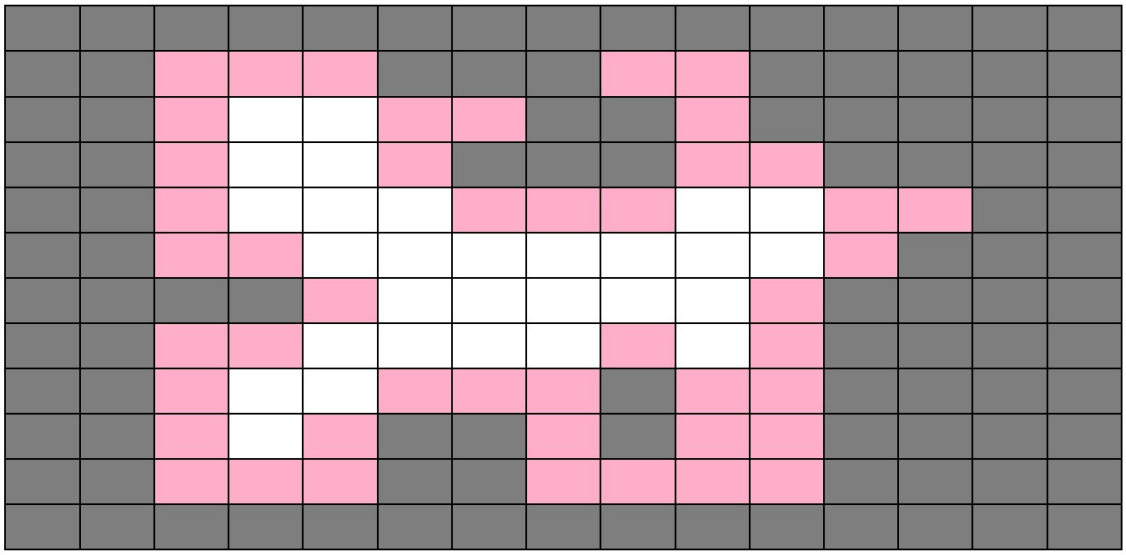
لبه چپ:



لبه پایین:



اجتماع چهار تصویر بالا را به دست می اوریم تا مرز تصویر حاصل شود(مرز های تصویر):



سوال ۶

الف) از فرمول های زیر که در اسلاید ها بود برای پیاده سازی استفاده کردم:

- عملگر گسترش (dilate) برای گسترش مجموعه A توسط B به صورت زیر تعریف می‌شود:

$$A \oplus B = \left\{ z \middle| (\hat{B})_z \cap A \neq \emptyset \right\}$$

anchor

- این رابطه به مفهوم بدست آوردن انعکاس B حول مرکز (لنگر) خودش و جابجایی آن به اندازه z است که اگر این نسخه از B دارای اشتراک با A بود، z جزء مجموعه جدید خواهد بود

- در یک تصویر سطح خاکستری، عملگر مورفولوژی گسترش به صورت زیر تعریف می‌شود

$$dst(x,y) = \max_{(x',y') \in SE} src(x+x', y+y')$$

- عملگر سایش (erode) برای فرسایش مجموعه A توسط B به صورت زیر تعریف می‌شود:

$$A \ominus B = \{z | (B)_z \subseteq A\}$$

- بنابراین سایش مجموعه A توسط B شامل مجموعه نقاطی است که به ازای آنها B به طور کامل درون A قرار می‌گیرد

- عملگر سایش برای تصاویر سطح خاکستری

$$dst(x,y) = \min_{(x',y') \in SE} src(x+x', y+y')$$

عملگر بسته

- عملگر بسته (closing) برای حذف حفره‌های کوچک و هموار کردن محیط نواحی تعریف شده است

$$A \cdot B = (A \oplus B) \ominus B$$

- این عملگر ناحیه‌های سیاه که در احاطه پیکسل‌های سفید هستند را حذف می‌کند

- عملگر باز (opening) برای حذف جزئیات کوچک و هموار کردن محیط نواحی تعریف شده است

$$A \circ B = (A \ominus B) \oplus B$$

- این عملگر ناحیه‌های سفید که در احاطه پیکسل‌های سیاه هستند را حذف می‌کند

برای گسترش ابتدا کرنل را ۱۸۰ درجه چرخاندم. سپس به اندازه نصف کرنل padding زدم از نوع reflect101 سپس با کانوالو کردن کرنل و max گرفتن به محاسبه پرداختم.
برای سایش نیز به اندازه نصف کرنل padding زدم از نوع reflect101 سپس با کانوالو کردن کرنل و min گرفتن از بین عناصری که در کرتل مقدار ۱ دارند به محاسبه پرداختم.
برای عملگر باز ابتدا سایش سپس گسترش خودم را صدا زدم.
برای عملگر بسته نیز ابتدا گسترش و سپس سایش دادم.
ب) از لینک زیر که به طور کامل توضیح داده استفاده کردیم و پیاده سازی کردیم:

https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html

حاصل با پیاده سازی های قسمت الف یکسان بود.

(امتیازی): از لینک زیر برای پیاده سازی کمک گرفتم:

https://medium.com/analytics-vidhya/skeletonization-in-python-using-opencv-b7fa16867331

توضیحات:

۱ .تابع **get_skeleton** تصویر ورودی و عنصر ساختاری **B** را به عنوان پارامترها دریافت می‌کند .
۲ .متغیر **res** به عنوان یک کپی از تصویر ورودی مقداردهی اولیه می‌شود.
۳ .تصویر ورودی با استفاده از **cv2.bitwise_not** معکوس می‌شود تا اشیاء سفید و پس زمینه سیاه شود.
۴ .تصویر معکوس شده با استفاده از مقدار آستانه ۱۲۷ به یک تصویر دودویی تبدیل می‌شود و نتیجه در **img** ذخیره می‌شود.
۵ .یک تصویر خالی به نام **skeleton** با ابعاد مشابه **img** ایجاد می‌شود تا اسکلت در آن ذخیره شود .
۶ .عنصر ساختاری **B** به عنوان پارامتر دریافت می‌شود .
۷ .یک حلقه آغاز می‌شود که تا زمانی که تصویر به طور کامل سایش شده و دیگر پیکسل‌های سفید باقی نمانده است، ادامه می‌یابد.
۸ .در هر تکرار از حلقه:

- تصویر با استفاده از تابع **open_morphology** و عنصر ساختاری **B** باز می‌شود و نتیجه در **open** ذخیره می‌شود.
- تفاوت بین تصویر اصلی و تصویر باز شده با استفاده از **cv2.subtract** محاسبه می‌شود و نتیجه در **temp** ذخیره می‌شود.

- تصویر اصلی با استفاده از تابع erode با عنصر ساختاری B فرسایش می یابد. نتیجه در فرسوده ذخیره می شود.
- تصویر موقت با استفاده از cv2.bitwise_or با تصویر اسکلت ترکیب شده و در اسکلت ذخیره می شود.
- تصویر فرسوده شده فرسوده شده برای تکرار بعدی در img کپی می شود.
- اگر هیچ پیکسل سفیدی در img باقی نماند که نشان می دهد تصویر کاملاً فرسایش یافته است، حلقه شکسته می شود.

۹. در نهایت، تصویر اسکلت به res اختصاص داده می شود.
۱۰. تصویر res حاصل که شامل اسکلت تصویر ورودی است، برگردانده می شود.

پایان