

رسالة محمد



مبانی بینایی کامپیوتر

مدرس: محمدرضا محمدی

۱۳۹۹

ناحيه بندی تصوير

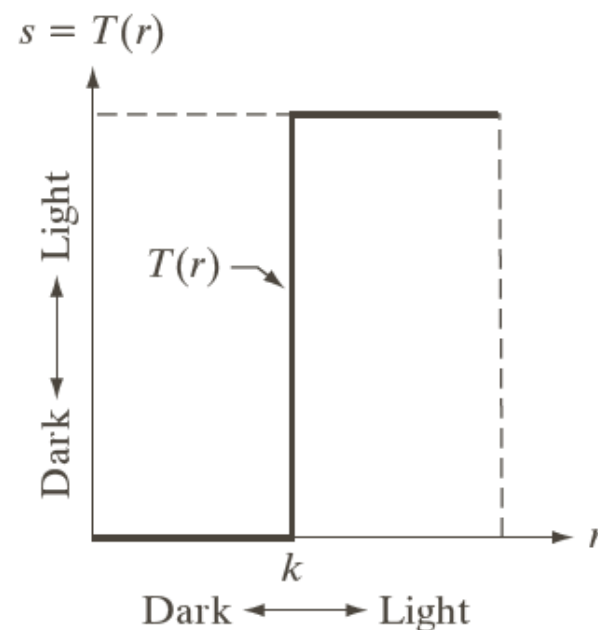
Image Segmentation

ناحیه بندی تصویر



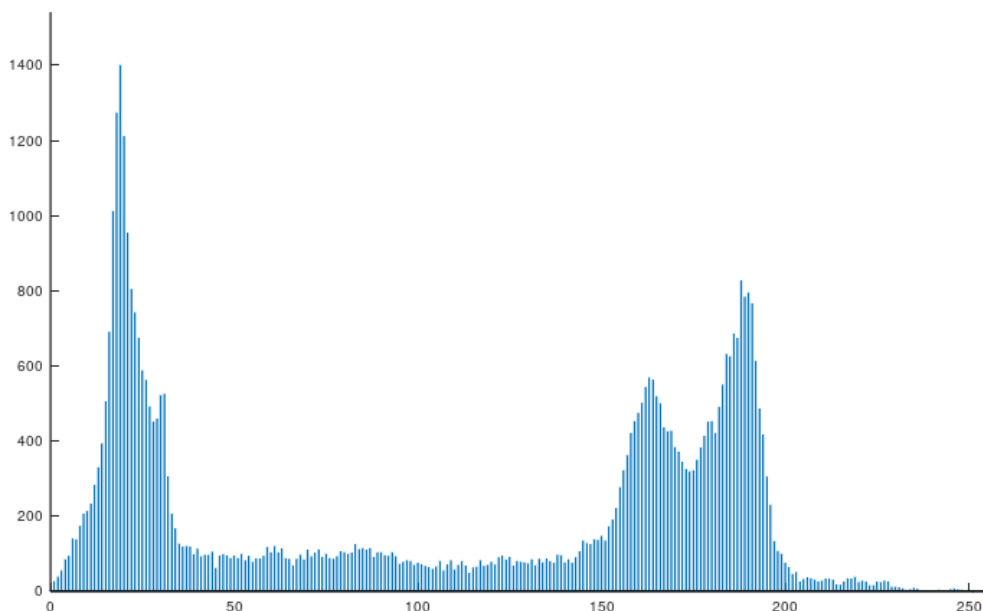
آستانه‌گذاری سطح خاکستری

- ساده‌ترین راه برای استخراج ناحیه از تصویر استفاده از مقادیر سطح خاکستری است
- پس از این عملگر نقطه‌ای، هر ناحیه به هم پیوسته یک ناحیه است



تعیین سطح آستانه

- سطح آستانه بهینه چه عددی است؟
- می‌توان با استفاده از دانش پیشین از یک عدد ثابت استفاده کرد
- می‌توان از مشخصه‌های آماری مانند میانگین یا میانه سطوح خاکستری استفاده کرد
- می‌توان از استفاده از هیستوگرام استفاده کرد



الگوریتم Otsu

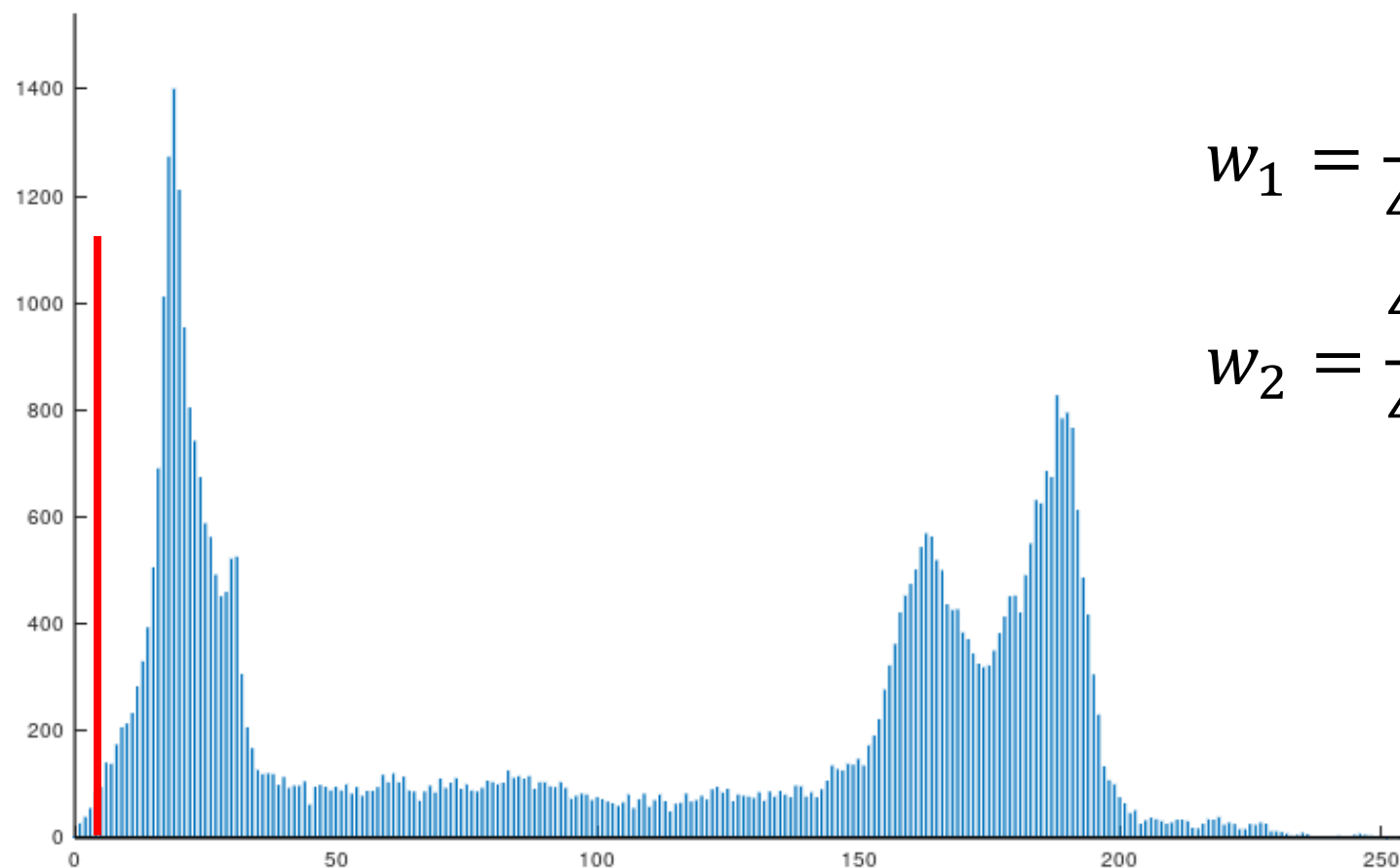
- یک الگوریتم تعیین سطح مقدار آستانه بر حسب مشخصه‌های آماری است
- خلاصه الگوریتم این است که سطح آستانه‌ای را انتخاب کنیم که واریانس بین پیکسل‌های هر کلاس کمینه شود

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

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

الگوریتم Otsu

- برای یک تصویر ۸ بیتی سطح آستانه یکی از ۲۵۵ مقدار است



$$w_1 = \frac{302}{48832} = 0.006$$

$$w_2 = \frac{48530}{48832} = 0.994$$

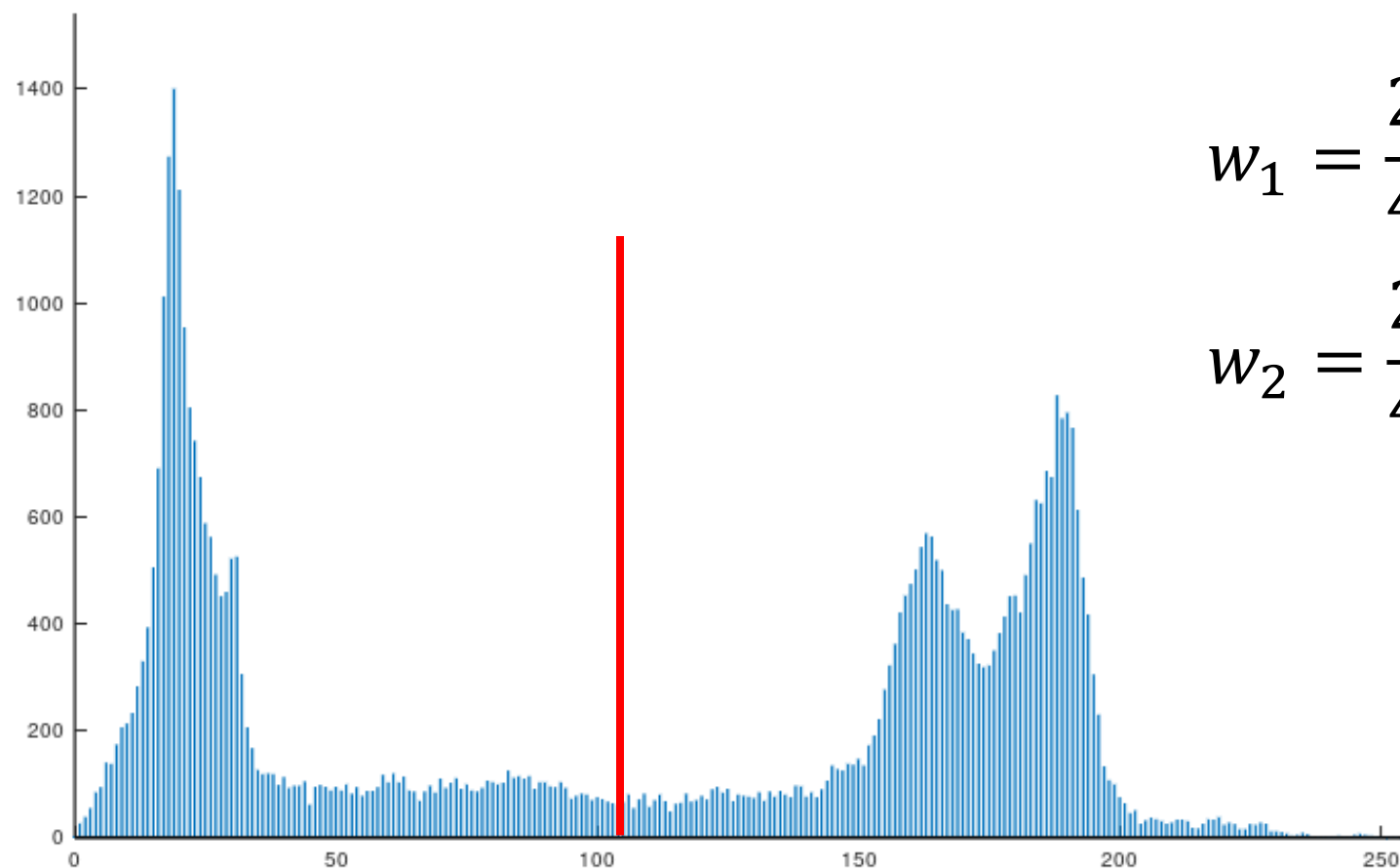
$$\sigma_1^2 = 1.75$$

$$\sigma_2^2 = 5154.8$$

$$\sigma_w^2 = 5123.0$$

الگوریتم Otsu

- برای یک تصویر ۸ بیتی سطح آستانه یکی از ۲۵۵ مقدار است



$$w_1 = \frac{23033}{48832} = 0.472$$

$$w_2 = \frac{25799}{48832} = 0.528$$

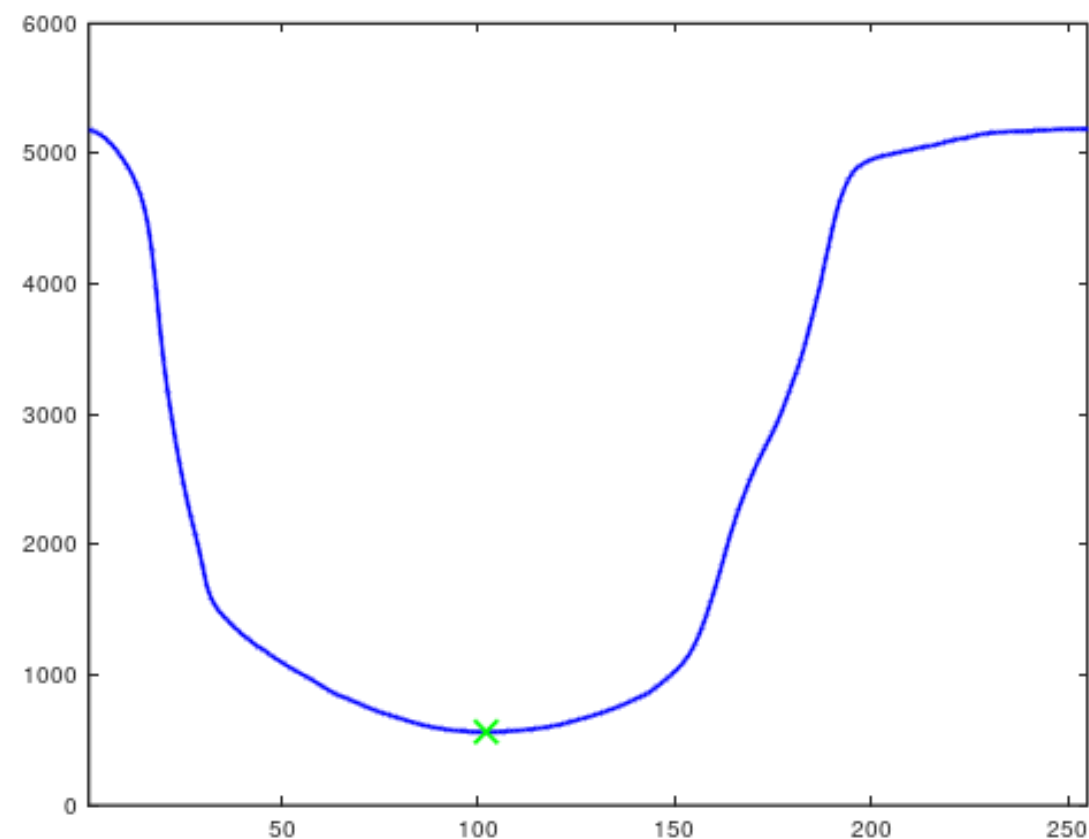
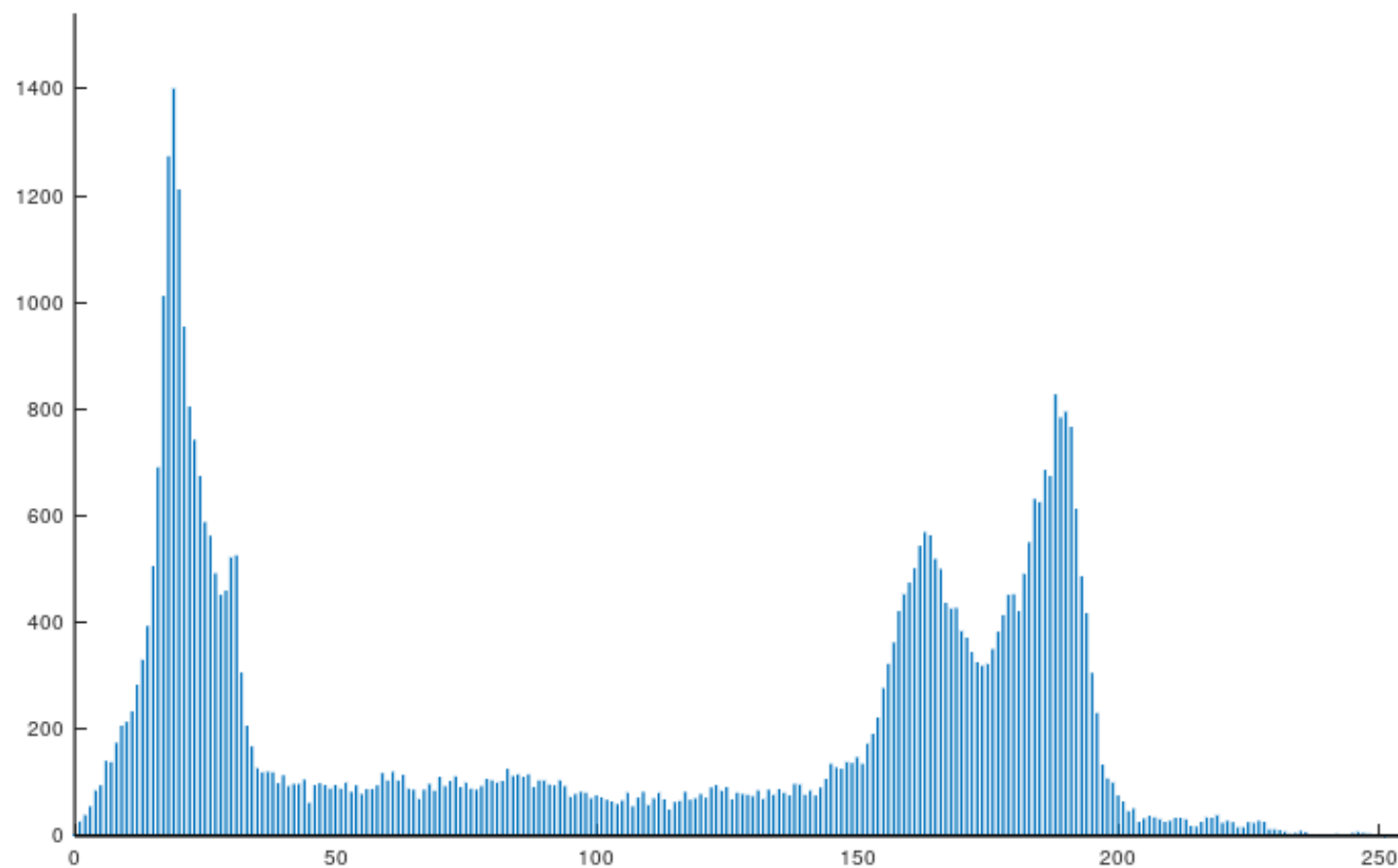
$$\sigma_1^2 = 703.2$$

$$\sigma_2^2 = 456.9$$

$$\sigma_w^2 = 573.1$$

الگوریتم Otsu

- برای یک تصویر ۸ بیتی سطح آستانه یکی از ۲۵۵ مقدار است



الگوریتم Otsu

1		8			6	9	2	
	2		4	9		1		
	6						4	5
		3		7				
	9					2		3
					5			9
9							8	
	5		1				6	4
		1		5				

1		8			6	9	2	
	2		4	9		1		
	6						4	5
		3		7				
	9					2		3
					5			9
9							8	
	5		1				6	4
		1		5				

blogs about all things funnies.

SUDOKU HARD

Answers to today's Sudoku and Scrabble Grams puzzles can be found Monday on the preceding page.

Fill in the blank spaces in the grid so that every vertical column, every horizontal row and every 3x3 box contains the numbers 1 through 9, without repeating any.

			6		4	7		
7		6						9
				5			8	
	7			2			9	3
8								5
4	3			1			7	
	5		2					
3							2	8
		2	3		1			

8-1-09

blogs about all things funnies.

SUDOKU HARD

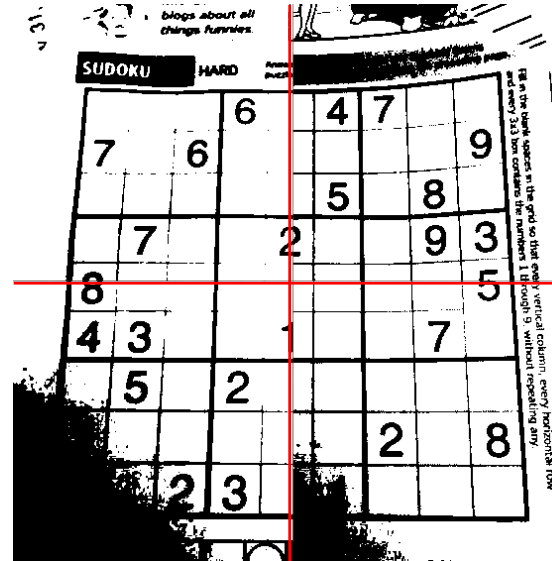
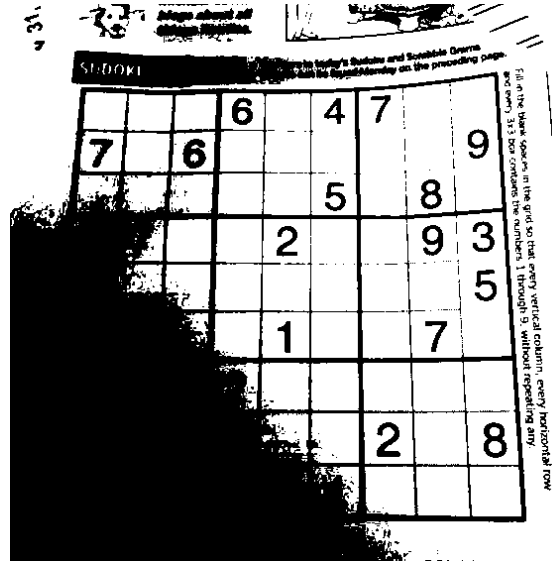
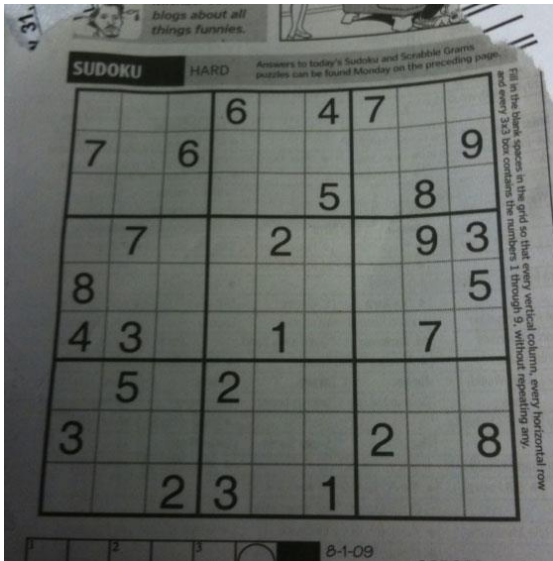
Answers to today's Sudoku and Scrabble Grams puzzles can be found Monday on the preceding page.

Fill in the blank spaces in the grid so that every vertical column, every horizontal row and every 3x3 box contains the numbers 1 through 9, without repeating any.

			6		4	7		
7		6						9
				5			8	
				2			9	3
								5
				1			7	
							2	8
		2	3		1			

آستانه گذاری افقی

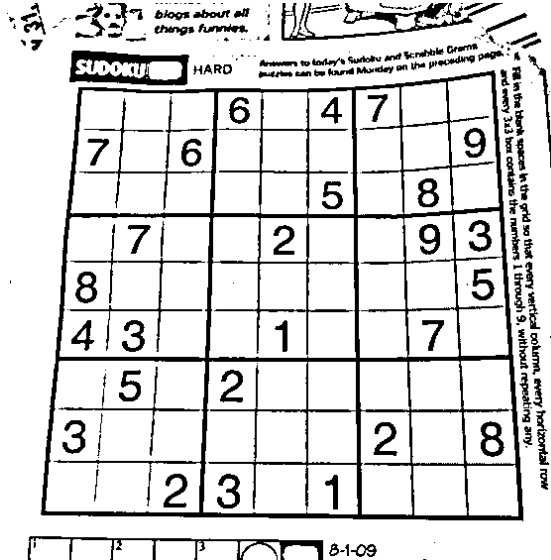
- به منظور رفع چالش قبل، مناسب است تا برای هر ناحیه از تصویر یک آستانه متناسب تعریف شود
- در حالت حدی می توان برای هر پیکسل یک آستانه تعریف کرد
- البته این محاسبات پیچیده برای هر پیکسل هزینه بر است
- می توان میانگین پیکسل های اطراف هر ناحیه را به عنوان معیاری برای مقدار آستانه محاسبه کرد



آستانه گذاری افقی

```
dst = cv2.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C)
```

// src:	Source 8-bit single-channel image
// maxValue:	Non-zero value assigned to the pixels for which the condition is satisfied
// adaptiveMethod:	Adaptive thresholding algorithm to use (MEAN or GAUSSIAN)
// thresholdType:	Thresholding type that must be either THRESH_BINARY or THRESH_BINARY_INV
// blockSize:	Size of a pixel neighborhood that is used to calculate a threshold value
// C:	Constant subtracted from the mean or weighted mean
// dst:	Destination image of the same size and the same type as src



استخراج ناحیه‌ها از تصویر باینری

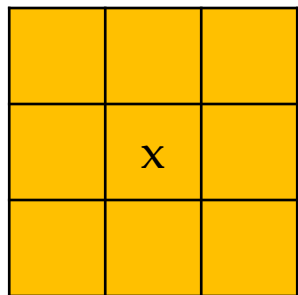
- با استفاده از روش‌های ذکر شده، یک تصویر دوسطحی بدست می‌آید
- حال باید پیکسل‌های مربوط به هر ناحیه را مشخص کنیم

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	0	1	0	1	0	0
0	1	0	0	0	0	0	1
1	1	0	0	0	1	1	1
1	1	0	0	1	1	0	0
1	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0

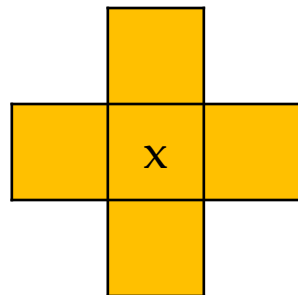
0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	0	1	0	1	0	0
0	2	0	0	0	0	0	3
2	2	0	0	0	3	3	3
2	2	0	0	3	3	0	0
2	0	0	0	0	0	3	3
0	0	0	0	0	0	3	0

اتصال پیکسل‌ها

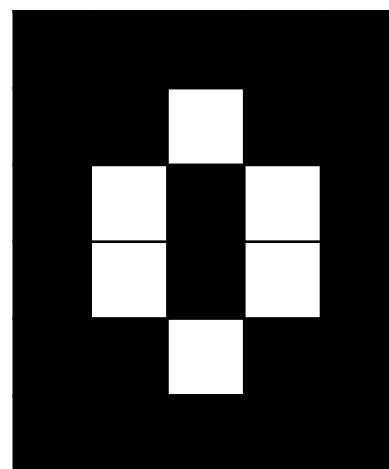
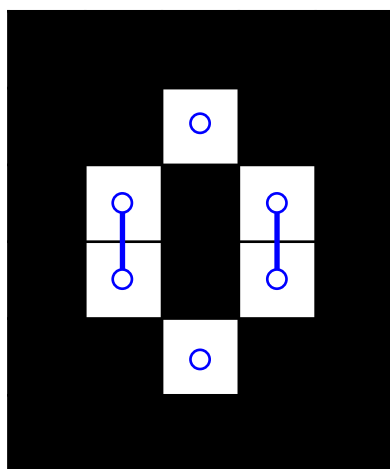
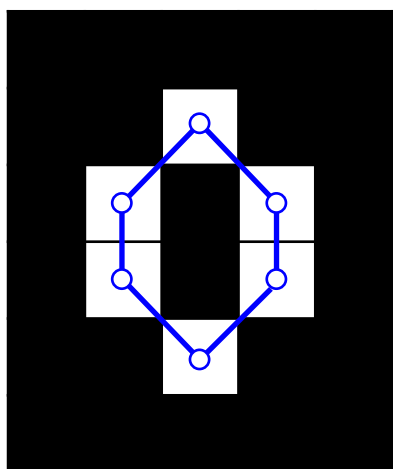
- کدام پیکسل‌ها به هم متصل هستند؟



8-connectivity

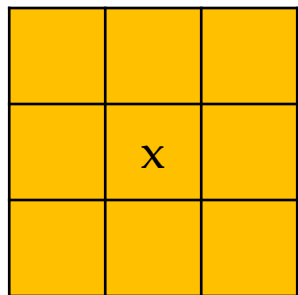


4-connectivity

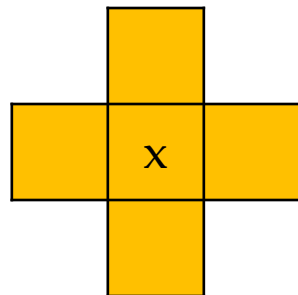


اتصال پیکسل‌ها

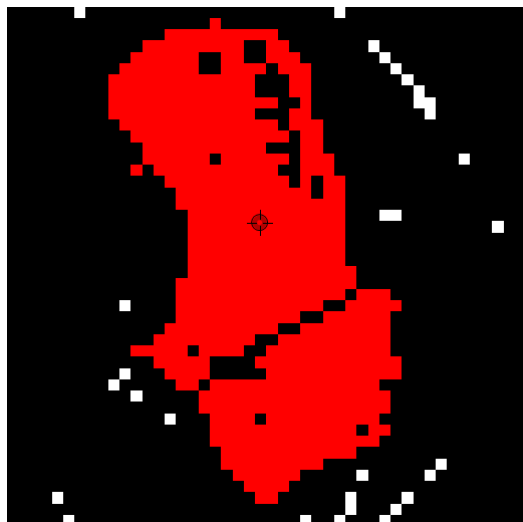
- کدام پیکسل‌ها به هم متصل هستند؟



8-connectivity

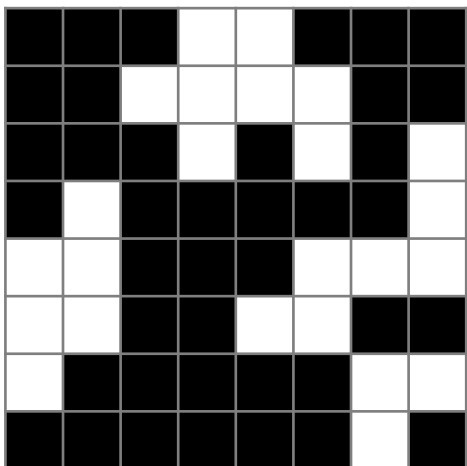


4-connectivity



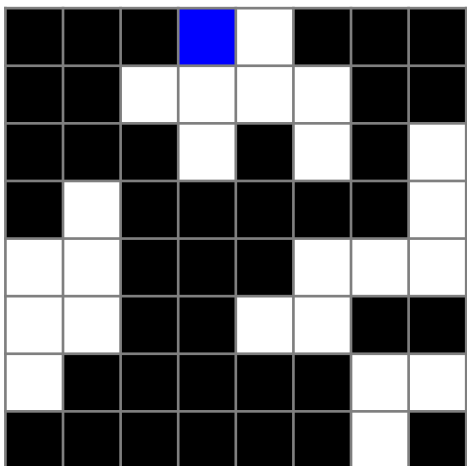
برچسب گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
 - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می کنیم و به هر پیکسل که رسیدیم پیکسل های متصل به آن را محاسبه می کنیم



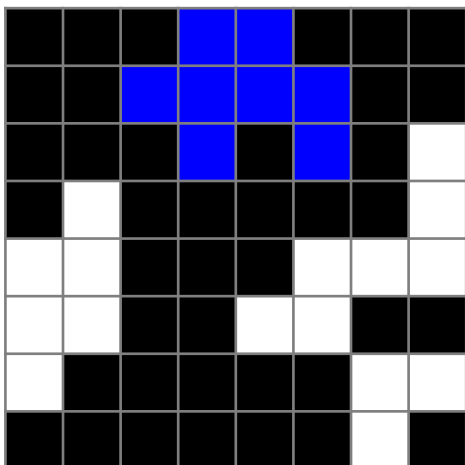
برچسب گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
 - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می کنیم و به هر پیکسل که رسیدیم پیکسل های متصل به آن را محاسبه می کنیم



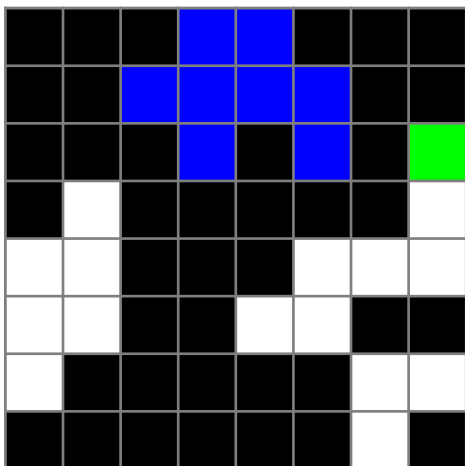
برچسب گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
 - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می کنیم و به هر پیکسل که رسیدیم پیکسل های متصل به آن را محاسبه می کنیم



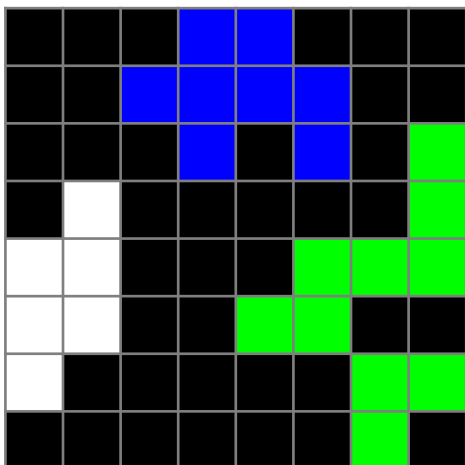
برچسب گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
 - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می کنیم و به هر پیکسل که رسیدیم پیکسل های متصل به آن را محاسبه می کنیم
 - سپس، این روند برای پیکسل های بعدی که هنوز برچسب نخورده اند تکرار می شود



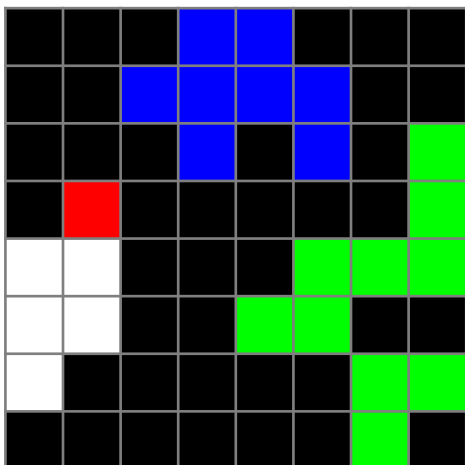
برچسب گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
 - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می کنیم و به هر پیکسل که رسیدیم پیکسل های متصل به آن را محاسبه می کنیم
 - سپس، این روند برای پیکسل های بعدی که هنوز برچسب نخورده اند تکرار می شود



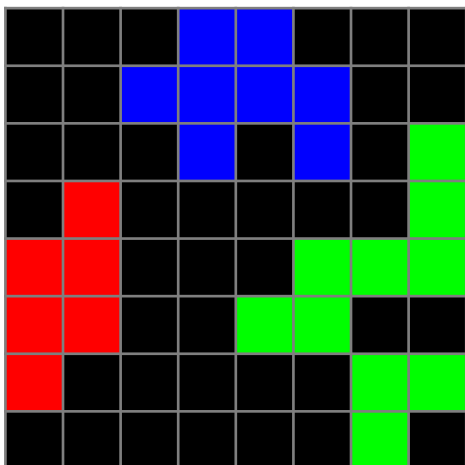
برچسب گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
 - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می کنیم و به هر پیکسل که رسیدیم پیکسل های متصل به آن را محاسبه می کنیم
 - سپس، این روند برای پیکسل های بعدی که هنوز برچسب نخورده اند تکرار می شود



برچسب گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
 - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می کنیم و به هر پیکسل که رسیدیم پیکسل های متصل به آن را محاسبه می کنیم
 - سپس، این روند برای پیکسل های بعدی که هنوز برچسب نخورده اند تکرار می شود



استخراج یک ناحیه متصل

```
// Finding the connected component containing an  
object pixel p
```

1. Initialize

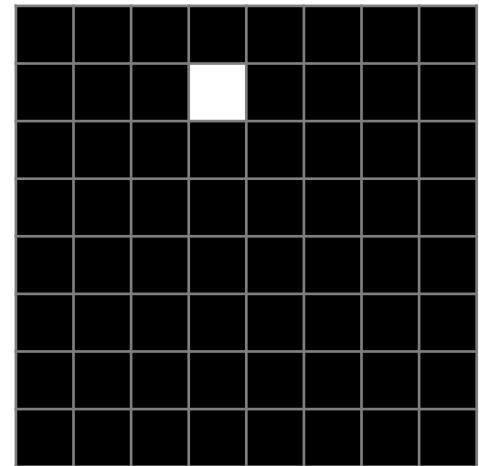
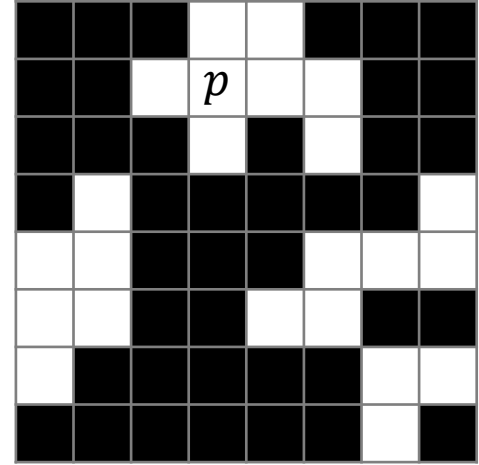
1. Create a result set S that contains only p
2. Create a Visited flag at each pixel, and set it to be False except for p
3. Initialize a queue (or stack) Q that contains only p .

2. Repeat until Q is empty:

1. Pop a pixel x from Q .

$$S = \{(1,3)\}$$

$$Q = \{(1,3)\}$$



استخراج یک ناحیه متصل

```
// Finding the connected component containing an  
object pixel p
```

1. Initialize

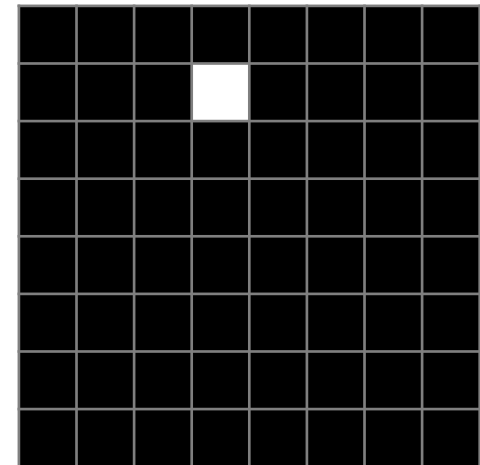
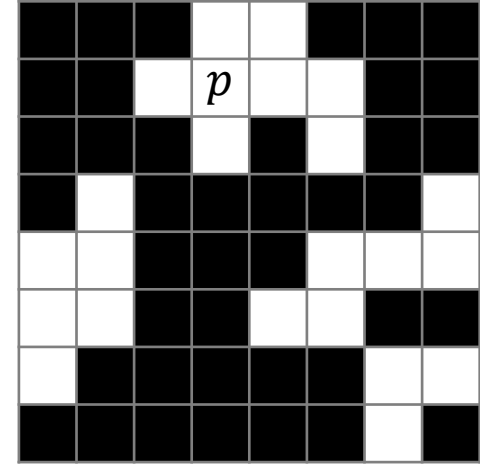
1. Create a result set S that contains only p
2. Create a Visited flag at each pixel, and set it to be False except for p
3. Initialize a queue (or stack) Q that contains only p .

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 .

$$S = \{(1,3)\}$$

$$Q = \{ \}$$



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 .

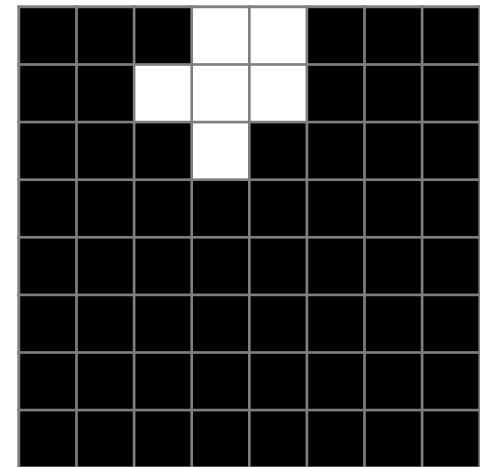
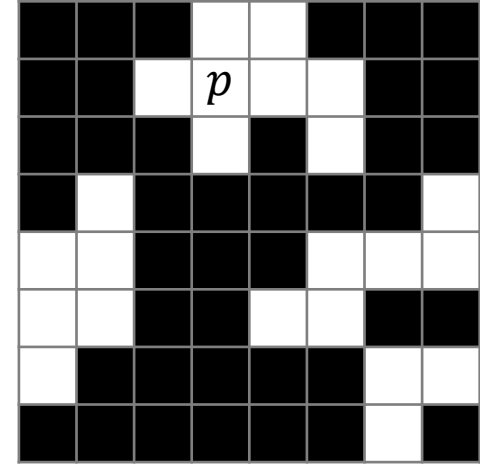
2. Repeat until Q is empty:

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

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

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



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 .

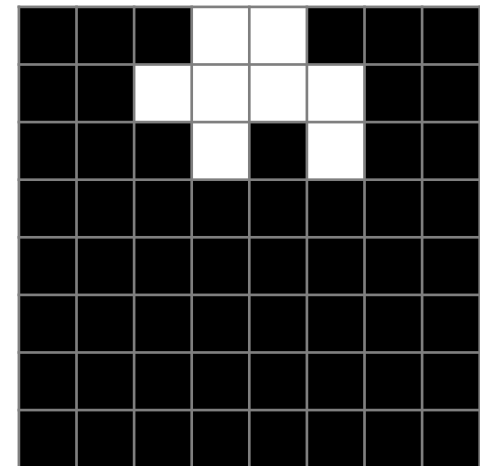
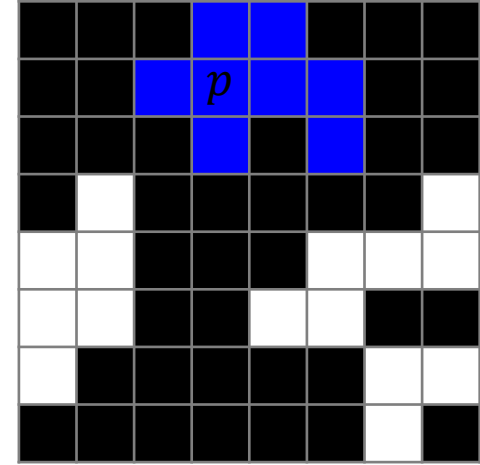
2. Repeat until Q is empty:

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

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

$$Q = \{ \}$$



استخراج یک ناحیه متصل

```
// Finding the connected component containing an  
object pixel p
```

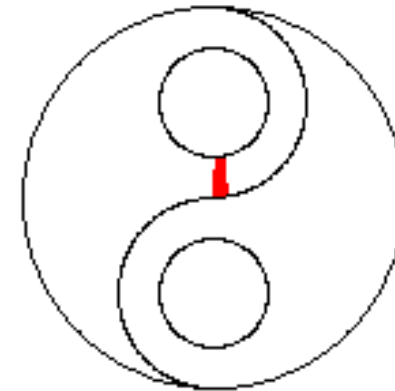
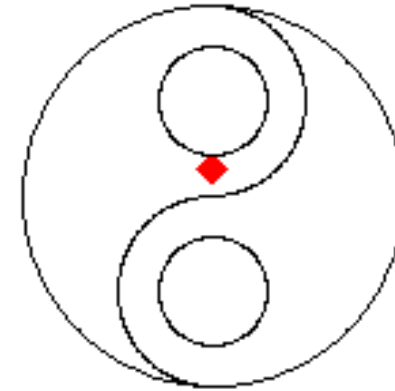
1. Initialize

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



رشد ناحیه

- هدف از این الگوریتم استخراج ناحیه مربوط به یک شیء در تصویر است که یک نقطه از آن را می دانیم
- به نقطه اولیه بذر یا seed گفته می شود



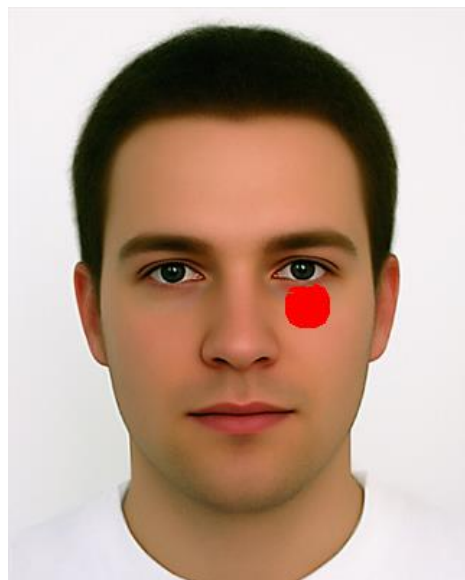
رشد ناحیه

- هدف از این الگوریتم استخراج ناحیه مربوط به یک شیء در تصویر است که یک نقطه از آن را می دانیم
- به نقطه اولیه بذر یا seed گفته می شود



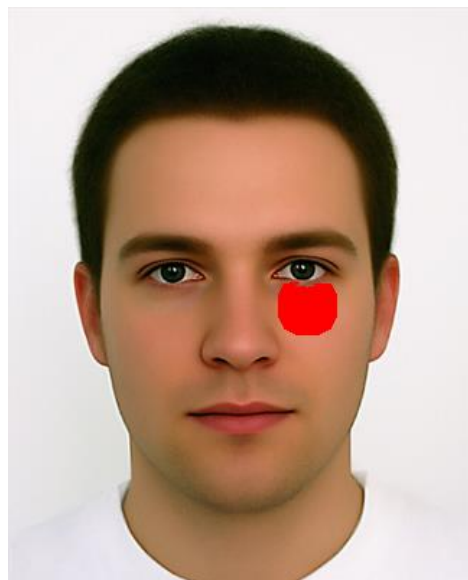
رشد ناحیه

- هدف از این الگوریتم استخراج ناحیه مربوط به یک شیء در تصویر است که یک نقطه از آن را می دانیم
- به نقطه اولیه بذر یا seed گفته می شود



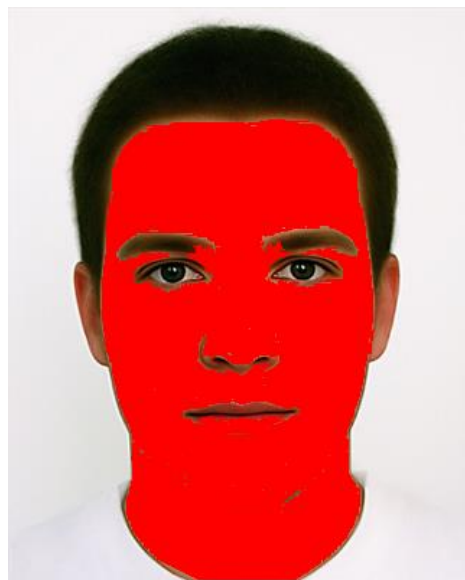
رشد ناحیه

- هدف از این الگوریتم استخراج ناحیه مربوط به یک شیء در تصویر است که یک نقطه از آن را می دانیم
- به نقطه اولیه بذر یا seed گفته می شود



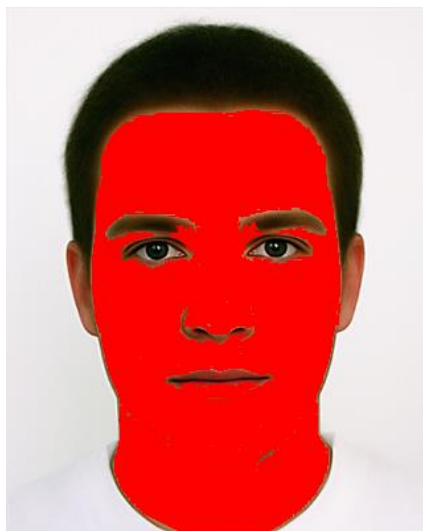
رشد ناحیه

- هدف از این الگوریتم استخراج ناحیه مربوط به یک شیء در تصویر است که یک نقطه از آن را می دانیم
- به نقطه اولیه بذر یا seed گفته می شود



رشد ناحیه

- الگوریتم رشد ناحیه مشابه با استخراج یک جزء متصل در تصویر باینری است
- تفاوت با تصویر باینری آن است که مقادیر پیکسل‌ها باینری نیستند و حتی می‌توانند رنگی باشند
- در پیاده‌سازی، تفاوت اصلی در این است که پیکسل‌های همسایه به چه شرطی به ناحیه اضافه شوند؟
- باید محتوای مشابهی داشته باشد که معادل با اختلاف کم است
- اختلاف با چه معیاری سنجیده شود؟



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

