پاسخ تمرین هشتم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

سوال اول) قسمت الف) Splitting and merging :

منبع: علاوه بر منبع معرفی شده کتاب Digital_image_processing_Gonzalez,_Rafael_C_Woods,_Richard_E بخش ۱۰,۴

ویدیو در یوتیوب: https://www.youtube.com/watch?v=ndHJmZqjJyM

Splitting and Merging Segmentation:

مرحله ی صفر: ابتدا روی کل تصویر شرط را اعمال میکنیم تا ببینیم کل تصویر یک ناحیه است یا خیر.

$Z_{max} = 7$
$Z_{min} = 0$
$T = 3$
$|Z_{max} - Z_{min}| <= 3$
$|7 - 0| > 3$

پس شرط شباهت بین پیکسل ها نقض شد.

مرحله یک: تصویر را به ۴ ناحیه ی جدا از هم تقسیم میکنیم:

۶	۴	۶	۶	۷	۷	۶	۶
۶	۷	۶	۷	۴	۴	۵	۷
۶	R1 ۶	۵	۵	۳	R2 ۲	۴	۶
۴	۵	۴	۵	۲	۳	۵	۶
۰	۳	۲	۳	۳	۲	۵	۷
۰	R3 ۰	۰	۰	۲	R4 ۲	۴	۶
۱	۱	۰	۱	۰	۳	۵	۵
۱	۰	۱	۰	۲	۳	۴	۵

مرحله ی دو: R1 را بررسی میکنیم:

۶	۴	۶	۶
۶	۷	۶	۷
۶	R1 ۶	۵	۵
۴	۵	۴	۵

$Z_{max} = 7$
$Z_{min} = 4$
$T = 3$
$|Z_{max} - Z_{min}| <= 3$
$|7 - 4| <= 3$

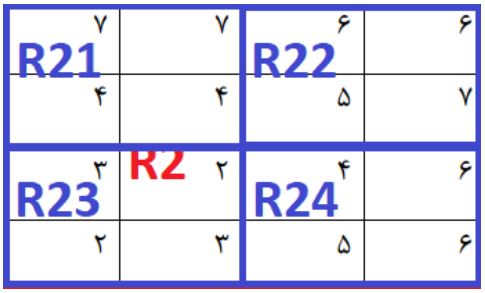
شرط شباهت برقرار است پس نیازی به split کردن این ناحیه نیست.

مرحله ی سه: R2 را بررسی میکنیم:

۷	۷	۶	۶
۴	۴	۵	۷
۳	R2 ۲	۴	۶
۲	۳	۵	۶

$Z_{max} = 7$
$Z_{min} = 2$
$T = 3$
$|Z_{max} - Z_{min}| <= 3$
$|7 - 2| = 5$

شرط شباهت پیکسل های این ناحیه برقرار نیست. پس این ناحیه را باید به ۴ قسمت تقسیم کرد.

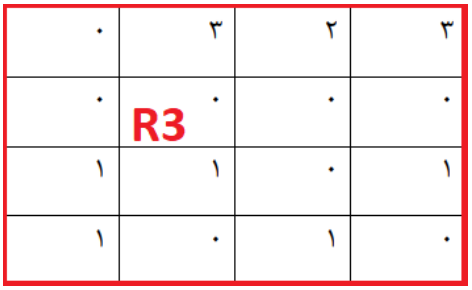


مرحله ی چهار: R2 را به چهار ناحیه تقسیم میکنیم:

سپس برای هر ناحیه مینیمم و ماکسیمم را مشخص میکنیم و اختلافشان را با حد آستانه مقایسه میکنیم.

$R21: Z_{max} = 7, Z_{min} = 4 => |7-4| <= 3$
$R22: Z_{max} = 7, Z_{min} = 5 => |7-5| <= 3$
$R23: Z_{max} = 3, Z_{min} = 2 => |3-2| <= 3$
$R24: Z_{max} = 6, Z_{min} = 5 => |6-5| <= 3$

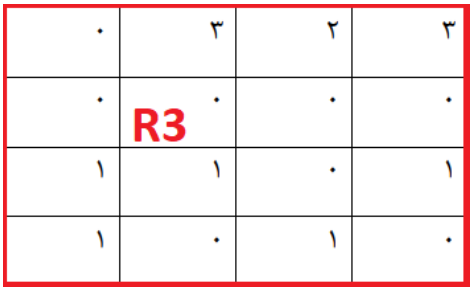
شرط مورد نظر روی پیکسل های هر یک از ناحیه ها برقرار است پس نیازی به split ندارند.



مرحله ی پنج:R3 را بررسی میکنیم:

$Z_{max} = 3$
$Z_{min} = 0$
$T = 3$
$|Z_{max} - Z_{min}| <= 3$
$|3-0| = 3$

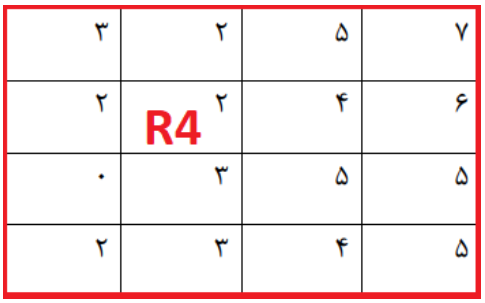
شرط برقرار است پس نیازی به split نیست.



مرحله ی پنج:R3 را بررسی میکنیم:

$Z_{max} = 3$
$Z_{min} = 0$
$T = 3$
$|Z_{max} - Z_{min}| <= 3$
$|3-0| = 3$

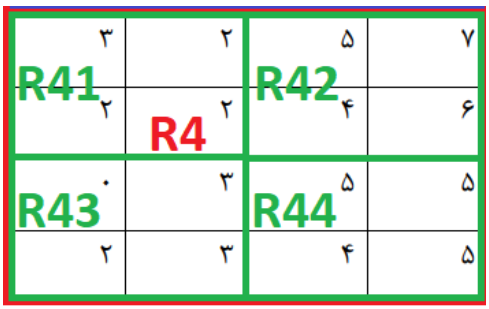
شرط برقرار است پس نیازی به split نیست.



مرحله ی شش: R4 را بررسی میکنیم:

$Z_{max} = 7$
$Z_{min} = 0$
$T = 3$
$|Z_{max} - Z_{min}| <= 3$
$|7-0| = 7$

شرط برقرار نیست پس باید این ناحیه به ۴ ناحیه ی جدا از هم تقسیم شود.



مرحله ی هفت: R4 را به چهار ناحیه تقسیم میکنیم:

سپس برای هر ناحیه مینیمم و ماکسیمم را مشخص میکنیم و اختلافشان را با حد آستانه مقایسه میکنیم.

$R41: Z_{max} = 3, Z_{min} = 2 => |3-2| <= 3$
$R42: Z_{max} = 7, Z_{min} = 4 => |7-4| <= 3$
$R43: Z_{max} = 3, Z_{min} = 0 => |3-0| <= 3$
$R24: Z_{max} = 5, Z_{min} = 4 => |5-4| <= 3$

شرط مورد نظر روی پیکسل های هر یک از ناحیه ها برقرار است پس نیازی به split ندارند.

پاسخ تمرین هشتم

مهدیه نادری: ۹۸۵۲۲۰۷۶

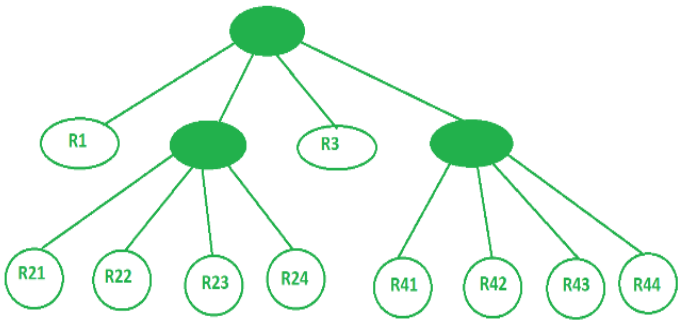
مرحله ی هشت: حالا ما ۱۰ ناحیه ی زیر را داریم:

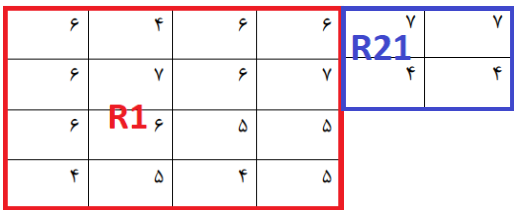
R1, R21, R22, R23, R24, R3, R41, R42, R43, R44

که این ناحیه ها اشتراکی با هم ندارند و بیشتر از این هم نمیشود به ناحیه های کوچکتر تقسیمشان کرد چون شرطی که تعیین کردیم برای همه برقرار شده است.

حالا نوبت Merging است.

باید همه ی ناحیه ها را طبق شرط با هم مقایسه کنیم. اگر در ترکیب این نواحدی شرط برقرار شد این ناحیه ها را ادغام میکنیم.



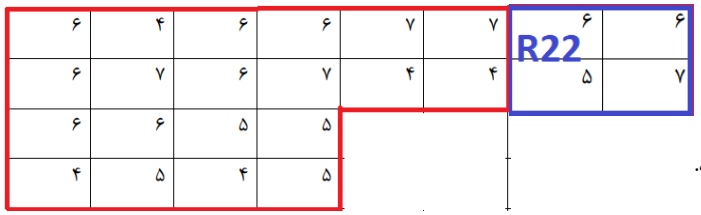


R1, R21:

$$Z_{max} = 7, Z_{min} = 4$$
$$|Z_{max} - Z_{min}| <= 3$$
$$|7 - 4| = 3$$

شرط لازم برقرار است پس این دو ناحیه را ادغام میکنیم.

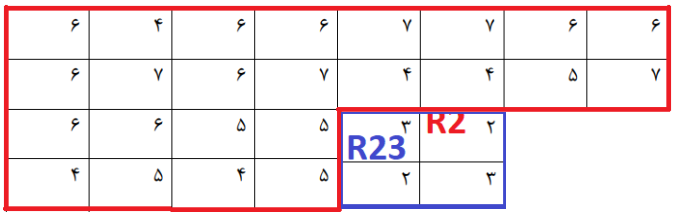
مرحله ی نه: R1R21و R22:



$$Z_{max} = 7, Z_{min} = 4$$
$$|Z_{max} - Z_{min}| <= 3$$
$$|7 - 4| = 3$$

پس این ناحیه ها هم چون شرط برقرار است با هم ادغام میشوند.

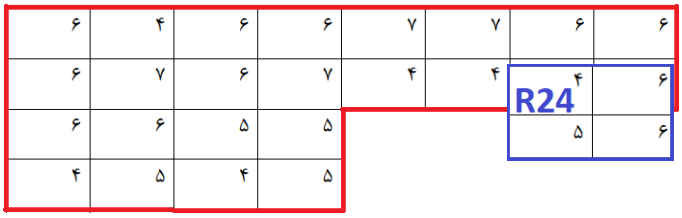
مرحله ی ده:R1R21R22 و R23:



$$Z_{max} = 7, Z_{min} = 2$$
$$|Z_{max} - Z_{min}| <= 3$$
$$|7 - 2| = 5$$

۵ از ۳ بزرگتر است. پس این دو ناحیه را با هم ادغام نمیکنیم.

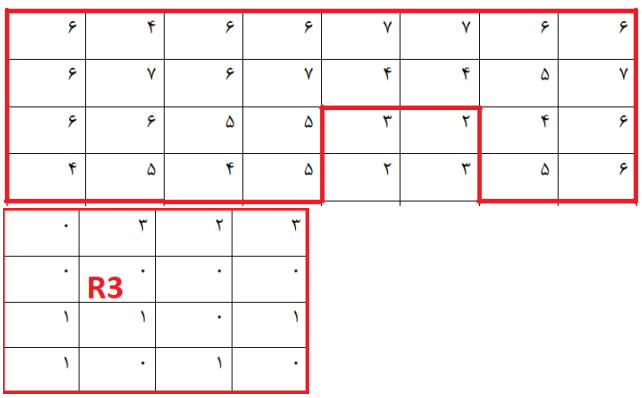
مرحله ی یازده: R1R21R22 و R24:



$$Z_{max} = 7, Z_{min} = 4$$
$$|Z_{max} - Z_{min}| <= 3$$
$$|7 - 4| = 3$$

شرط ادغام برقرار است پس با هم ادغام میشوند.

مرحله ی دوازده :R1R21R22R24 و R3:



$$Z_{max} = 7, Z_{min} = 0$$
$$|Z_{max} - Z_{min}| <= 3$$
$$|7 - 0| = 0$$

شرط ادغام برقرار نیست پس ادغام نمیشوند.

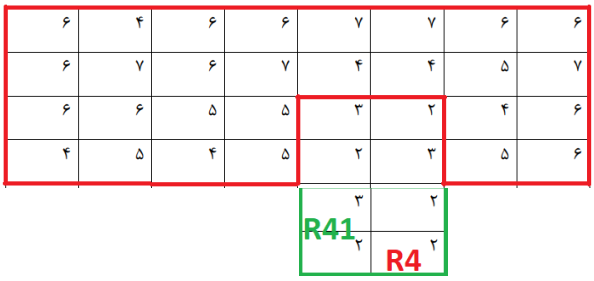
پاسخ تمرین هشتم

مهدیه نادری: ۹۸۵۲۲۰۷۶

مرحله ی سیزدهم :R1R21R22R24 و R41:

$$Z_{max} = 7, Z_{min} = 2$$
$$|Z_{max} - Z_{min}| <= 3$$
$$|7 - 2| = 5$$

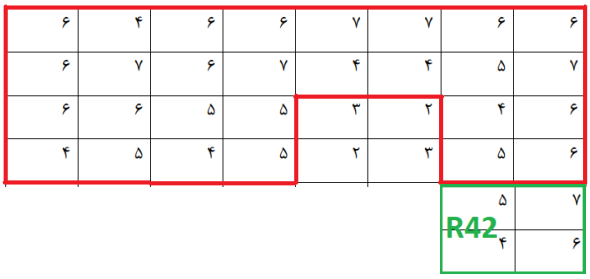
شرط ادغام برقرار نیست پس ادغام نمیشوند.



مرحله ی چهاردهم :R1R21R22R24 و R42:

$$Z_{max} = 7, Z_{min} = 4$$
$$|Z_{max} - Z_{min}| <= 3$$
$$|7 - 4| = 3$$

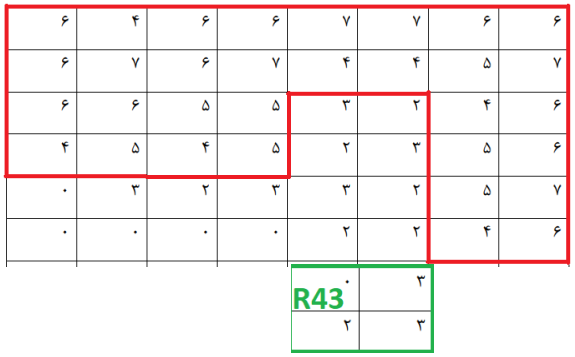
شرط ادغام برقرار است پس ادغام میشوند.



مرحله ی پانزدهم: R1R21R22R24R42و R43:

$$Z_{max} = 7, Z_{min} = 0$$
$$|Z_{max} - Z_{min}| <= 3$$
$$|7 - 0| = 0$$

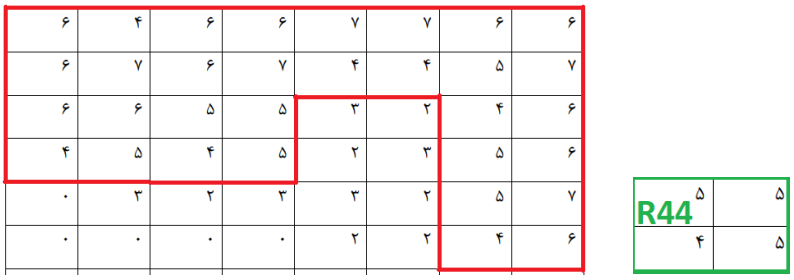
شرط ادغام برقرار نیست پس ادغام نمیشوند.



مرحله ی شانزدهم: R1R21R22R24R42و R44:

$$Z_{max} = 7, Z_{min} = 4$$
$$|Z_{max} - Z_{min}| <= 3$$
$$|7 - 4| = 3$$

شرط ادغام برقرار است و ادغام میشوند.



دیگر ناحیه ای برای مقایسه نیست.

نواحی باقی مانده با هم یک Segment را تشکیل میدهند چون اصولا کار آستانه گذاری همین است. برای اطمینان درستی جواب شرط بین آنها را هم در زیر بررسی کردیم.

R3و R23:

$$Z_{max} = 3, Z_{min} = 0, |3 - 0| = 3$$

R3R23و R41:

$$Z_{max} = 3, Z_{min} = 0, |3 - 0| = 3$$

R3R23R41و R43:

$$Z_{max} = 3, Z_{min} = 0, |3 - 0| = 3$$

همگی با R3 ادغام شدند.

نتیجه ی زیر حاصل میشود که در آن دو ناحیه در تصویر پیدا کردیم.

۶	۴	۶	۶	۷	۷	۶	۶
۶	۷	۶	۷	۴	۴	۵	۷
۶	۶	۵	۵	۳	۲	۴	۶
۴	۵	۴	۵	۲	۳	۵	۶
۰	۳	۲	۳	۳	۲	۵	۷
۰	۰	۰	۰	۲	۲	۴	۶
۱	۱	۰	۱	۰	۳	۵	۵
۱	۰	۱	۰	۲	۳	۴	۵

سوال اول) قسمت ب)مقایسه ی Splitting and merging و Region Growing:

رویکرد Spliting and merging:

برعکس رویکرد رشد ناحیه است.

ایده ی اولیه : شباهت پیکسل های تصویر با یک روش شباهت سنجی بررسی میشود اگر شبیه نبودند به ۴ ناحیه ی کاملا جدا از هم بدون نقطه ی مشترک تقسیم میشود. بعد از این میایم ۴ ناحیه ی ایجاد شده رو با یکدیگر مقایسه میکنیم که چه قدر به هم شبیه هستند. ناحیه هایی که پیکسل هایش شبیه هم بودند را نیازی نیست split کنیم و اگر ناحیه ای متفاوت بود آن را Split میکنیم و همین سنجش شباهت را انجام میدهیم تا جایی که یا همه ی ۴ زیر ناحیه مشابه باشند یا هم دیگر امکان split نباشد.

Split any Rj rigion t 4 region for which Q(Rj) = False until splitting is possible.

زمانی که دیگر هیچ تقسیم و جداسازی ای در ناحیه ها ممکن نبود merging شروع میشود. از یک ناحیه شروع میکنیم و با ناحیه های دیگر از نظر شباهت مقایسه میکنیم. اگر شرط تشابه برقرار بود با هم merge میشوند. زمانی متوقف میشویم که دیگر merging ممکن نباشد.

Merge any adjacent region Rj and Rk which $Q(R_j \cup R_k) = \text{TRUE}$ until merging is possible

یک الگوریتم بالا به پایین محسوب میشود.

رویکرد رشد ناحیه:

در الگوریتم رشد ناحیه ما یک تصویر رنگی داریم و میخواهیم ناحیه ی رنگی بدست بیاریم. اما راحت نیست ترشلد بگذاریم برای رنگ ها.

به جای آن میخواهیم از یک ناحیه شروع کنیم و کل شی رو در بیاریم.

گاهی این نقطه دستی پیدا میشه و گاهی اتوماتیک. برای اتوماتیک باید یک مشخصه ای از آن بدانیم. به این نقطه میگیم بذر یا seed.

در اینجا معمولا از صف استفاده میشه. یعنی اول اطرافش بررسی میشه بعد نقاط دور تر اما میشود از پشته هم استفاده کرد.

الگوریتم رشد ناحیه مشابه با استخراج یک جزء متصل در تصویر باینری است اما اینجا تصویر باینری نیست که بگیم نقطه اگه مثلا یک بود متصل بشود. به جای آن باید مشابهت بسنجیم. مثلا این پیکسل چه قدر به ناحیه ی ماشبیه است. اضافه بشه بهتره یا نه؟

یعنی یک مسئله ی کلاس بندی است که اضافه شدن نقطه ناحیه ی فعلی رو بهبود میده یا نه. شبیه هست یا نه؟

نکته ی مهم در پیاده سازی: شرط اضافه شدن پیکسل به ناحیه:

معیار اختلاف برای رشد ناحیه:

روش اول:

راحت ترین روش مقایسه با پیکسل بذره. اگه اختلاف کمتر از یک حدی بود به ناحیه اضافه بشه معادل با اینکه تصویر را بر اساس اختلاف با رنگ مورد نظر باینری کرده سپس ناحیه متصل به این پیکسل را استخراج کنیم. باینری کردن بر اساس شباهت اتفاق میفته.

روش دوم:

پاسخ تمرین هشتم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

به جای مقایسه با پیکسل بذربا یک پیکسل نزدیک خودش مقایسه میشه. یعنی به تدریج معیار مقایسه عوض میشه. به پیکسل های مرزی توجه میکنه.

روش سوم:

بعضی مواقع هر دو روش قبل را در نظر میگیرن و ترکیب میکنن. یعنی هم با همسایه ها و هم با بذر مقایسه میکنند.

الگوریتم:

1. Find all connected components in $S(x, y)$ and reduce each connected component to one pixel; label all such pixels found as 1. All other pixels in S are labeled 0.
2. Form an image f_Q such that, at each point (x, y), $f_Q(x,y) = 1$ if the input image satisfies a given predicate, Q, at those coordinates, and $f_Q(x,y) = 0$ otherwise.
3. Let g be an image formed by appending to each seed point in S all the 1-valued points in f_Q that are 8-connected to that seed point.
4. Label each connected component in g with a different region label (e.g.,integers or letters). This is the segmented image obtained by region growing.

مقایسه ی دو رویکرد:

منبع:https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic3.htm

ویژگی های ناحیه بندی خوب:

۱-درنهایت همه ی پیکسل های تصویر به یک ناحیه مربوط شوند.

۲-هر پیکسل فقط در یک ناحیه معرفی شود.

۳-هر ناحیه مجموعه ای از پیکسل های به هم متصل باشد.

۴-هر ناحیه با توجه به یک تابع مشخص تعیین شده یکنواخت باشد.

۵-هر جفت ادغام شده از ناحیه های مجاور باید نسبت به تابع داده شده غیر یکنواخت باشد.

شباهت:

هر دو روش region-based هستند. یعن به جای انکه اول مرز ناحیه ها را پیدا کنند بعد خودشان را، همان اول از نقاط موجود در ناحیه بررسی را شرروع میکنند.

هر دو روش را میتوان برای تصاویر رنگی و پیدا کردن اجزای تصویر رنگی استفاده کرد.

هر دو میتوانند برای تصویر نویزی نسبتا خوب عمل کنند.

در هر دو به یک تابع یا روش برای محاسبه ی شباهت نواحی یا پیکسل ها نیاز داریم.(در روش های مختص باینری آستانه گذاری کافی بود)

هر دو روش iterative و تکرار شونده هستند.

تفاوت:

۱-درنهایت همه ی پیکسل های تصویر به یک ناحیه مربوط شوند.

۲-هر پیکسل فقط در یک ناحیه معرفی شود.

۳-هر ناحیه مجموعه ای از پیکسل های به هم متصل باشد.

۴-هر ناحیه با توجه به یک تابع مشخص تعیین شده یکنواخت باشد.

۵-هر جفت ادغام شده از ناحیه های مجاور باید نسبت به تابع داده شده غیر یکنواخت باشد.

در رشد ناحیه شرط ۳و۴ برقرارندو شرط ۱و۲ نیستند و شرط ۵ ممکن است نباشد.

در جداسازی و ادغام شرط ۲و۴و۵ حتما برقرارند و ۱و۳ ممکن است برقرار نشوند.

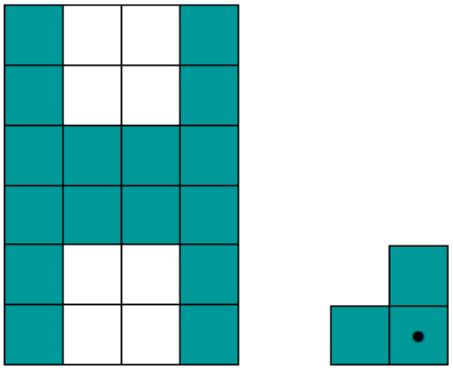
الگوریتم جدا سازی و ادغام نیازی به پیدا کردن نقطه ی بذر ندارد. و شروع آن به جای یک نقطه از یک ناحیه ی بزرگ است.

رشد ناحیه نمیتواند پیکسل هایی که به طور مستقیم یا غیر مستقیم در مجاورت هم نیستند را به عنوان یک ناحیه معرفی کند اما جداسازی ادغام دو ناحیه ی غیر مجاور که مشابه باشند را هم میتواند با یک برچسب نشان دهد.

پاسخ تمرین هشتم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

اگر در رشد ناحیه همیشه شباهت پیکسل ها با بذر سنجیده شود این الگوریتم سراسری و الگوریتم جدا سازی و ادغام محلی محسوب میشود.
چون ابتدا بیشترین ناحیه های ممکن در تصویر را پیدا میکند بعد شباهت آنها را با هم برای برچسب گذاری نهایی میسنجد.
رشد ناحیه پایین به بالا و دیگری الگوریتم بالا به پایین است.

سوال دوم) تفاوت بین اجرای عملگر باز را برای 1 یا 2 بار بنویسید. سپس، نتیجه 2 بار اجرای این عملگر را با توجه عنصر ساختاری و تصویر داده شده، ترسیم کنید:



تفاوت بین اجرای عملگر باز برای ۱ یا ۲ بار:

در اجرای اول تعدادی از پیکسل ها که شامل جزئیات بودند مثلا طول و عرض آن ها در چپ و پایین کمتر از یک بود حذف شدند یا به بیان دیگر نواحی سبز کوچکتر از عنصر ساختاری حذف شدند.. سپس آن را گسترش دادیم یعنی نواحی باقی مانده smooth شده و رشد کرد.. تصویر اولیه با تصویر حاصل تفاوت دارد.

تعریف دیگر عملگر باز $A \circ B = \cup\{(B)_z | (B)_z \subseteq A\}$ یعنی اجتماع تمام مجموعه نقطه های انتقال یافته ی B که زیر مجموعه ی A هستند یعنی به جای دو مرحله ای حساب کردن و پیچیده کردن کار میتوانیم عنصر ساختاری را روی صفحه ی تصویر بچرخانیم جاهایی که کاملا منطبق هستند را سبز کنیم، دقت کنید که فقط مرکز را نه بلکه هر سه خانه که با A مشترک هستند. نتیجه همین خواهد شد و چون در یک مرحله انجام میشود احتمال خطا در آن کمتر است.

وقتی دوباره عملگر باز را روی حاصل اعمال میکنیم نتیجه ی بدست آمده همان خواهد شد. در واقع هر چند بار که تکرار کنیم نتیجه عوض نمیشود چون با یکبار حساب کردن تمام نقاط مشترک را بدست آوردیم و نقطه ای نیست که خودش و همسایه هایش مشابه عنصرساختاری باشند و زیر مجموعه ی A باشد و در حاصل نیامده باشد. پس هرچه قدر بخواهیم میتوانیم امتحان کنیم.

عملگر بسته هم این ویژگی را دارد.

در شکل زیر در هر مرحله بخشی که کادر قرمز دور آن است تصویر اصلی و پیکسل های اطرافش برای padding هستند.

موضوع:

سؤال ۲: عملگر باز (opening): حذف جزئیات کوچک و هموار کردن حاشیه نواحی

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

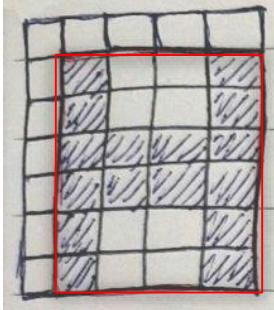
erode یا سایش — dilate یا گسترش

ناحیه های سفید که در احاطه‌ی پیکسل های سیاه هستند را حذف می‌کند.

باید ابتدا عمل فرسایش را با عنصر ساختاری B روی A اعمال کنیم.

سپس از آن در جاهایی که لازم است padding تصویر را با پیکسل های سفید پر کردیم.

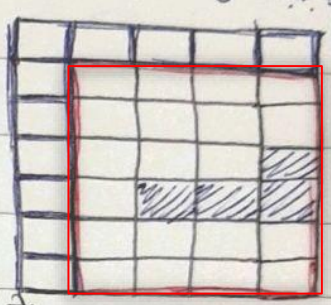
با توجه به شکل عنصر ساختاری پایین و سمت راست تصویر نیاز به padding ندارد.



عنصر ساختار

مرحله ۱: سایش 2D

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

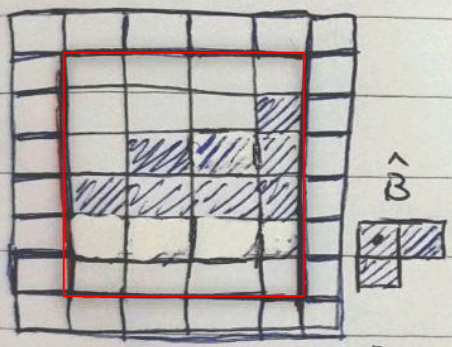


مرحله ۲: گسترش 2D

$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$ for $b \in B$

$\hat{B} = \{w \mid w = -b\}$

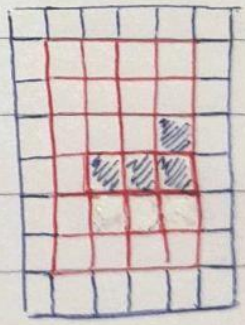
نیاز به padding در راست و پایین $\hat{B}$ داریم بخاطر شکل



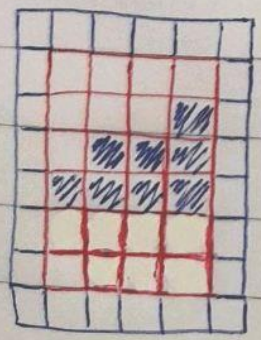
$\hat{B}$

(پایان ۱ بار اجرا عملگر باز)

مرحله ۱: سایش 2D با B



مرحله ۲: گسترش 2D با B



پایان ۲ بار اجرای عملگر باز

پاسخ تمرین هشتم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

سوال سوم) الف: نقاط ضعف و قوت الگوریتم Otsu و Adaptive Treshold (از نظر سرعت و عملکرد):

منبع: جلسه ی ۱۴

همانطور که میدانیم ما برای ناحیه بندی تصویر نیاز به آستانه گذاری روی آن داشتیم تا ابتدا با باینری کردن تصویر بک گراند تصویر را از اجزا جدا کنیم. سپس اجزاء را از هم تشخیص دهیم. مثلا پیدا کردن خطوط و اعداد جدول.

برای اینکار میتوانیم یک آستانه برای کل یک تصویر انتخب کنیم. که به آن Global threshlding میگویند و در حقیقت Otsu همین کار را میکند اما با مراحل خاصی.

Otsu: الگوریتم تعیین سطح آستانه بر حسب مشخصه های آماری تصویر

سطح آستانه ای رو انتخاب کنیم که واریانس بین پیکسل های هر کلاس مینیمم بشود. (برای باینری کردن دو کلاس داریم یعنی این سطح نقاط را به دو دسته تقسیم میکند)

اول حساب میکند چند درصد از پیکسل های تصویر متعلق به گروه یک و چند درصد متعلق به گروه دو هستن.

واریانس یک گروه نشون میدهد شدت روشنایی این پیکسل ها نسبت به میانگینشان چه قدر فاصله دارند.

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

پیکسل هایی که پایین تر از ترشلد باشن گروه یک و پیکسل های بالا تر از سطح آستانه در گروه دو قرار میگیرند.

حالت ایده آل صفر شدن مقدار واریانس است پس باید حالتی که تابع بالا مینیمم باشد را حساب کنیم.

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

منظور از واریانس پیکسل ها ، واریانس شدت روشنایی یا شدت رنگ پیکسل ها است.

برای مینیمم کردن تابع باید مشتق تابع نسبت به پارامتر سطح آستانه را حساب کنیم. اینجا نمیشه مشتق گرفت. مثلا ترشلد بین ۱۲۵ و ۱۲۶ خیلی فرقی نمیکنه. تابع خوش تعریفی برای مشتق گرفتن نیست.

یه کار ساده تر:

ترشلد ما ۲۵۵ تا مقدار مختلف میتونه داشته باشه: بین ۰و ۱- بین ۱و۲- بین ۲و۳ -.....-بین ۲۵۵و ۲۵۶

وقتی تعداد نقاط کمه میشود همه ی حالت ها را خیلی ساده تست کرد.

برای تصویر ۸ بیتی ۲۵۵ مقدار داریم. محاسبه ی فرمول ها برای ۲۵۵ پیکسل خیلی طولانی نیست و اصلا تابعی از ابعاد تصویر نیست. محاسبه ی هیستوگرام البته یک مقدار طول میکشد. اما بعد از بدست آوردن هیستگرام بقیه ی محاسبات روی همان هیستوگرام است و کاری با ابعاد پیکسل ندارد که چند مگا پیکسل است. یعنی ۲۵۵ محاسبه ی سبک داریم که میشه هر ۲۵۵ تا را سریع انجام داد و بهترین رو انتخاب کرد.

این الگوریتم روی تصویر همراه با سایه از جدول اعداد اصلا خوب اجرا نشد. چون سایه باعث شد یک سری خانه های جدول تیره تر از سطح آستانه بشوند و کلا سیاهشان کرد. کلا از ۰ تا ۲۵۵ همه ی حالت ها ی ترشولد رو کشیدیم اولاش ناحیه بندی جدول خوب بود ولی جزئیات و نوشته هاش سفید بودن بعد هم که جزئیات بهتر شد نصف تصویر تقریبا هی سیاه میشد.

بهتر است برای تصاویر نویزی و یا با کنتراست و شدت نور محیط نا مناسب از ترشلد اداپتیو استفاده بشه یعنی با قسمت تیره یه جور برخورد کنه با قسمتای روشن یه جور برخورد کنه. این ایده مطرح میشه که حد آستانه رو سراسری درنظر نگیریم یعنی برای کل تصویر یک حد آستانه در نظر نگیریم.

حل چالشOtsu: تعریف یک حد آستانه برای هر ناحیه ی تصویر. قرار نیست کل تصویر با یک ترشلد انجام بشه. بهتره محلی نگاه کنه و بگه مثلا یک پیکسل نسبت به اطراف خودش تیره تره. میشه روی هرپیکسل اتسو اجرا کرد و سطح آستانه ی مختص به خودش رو پیدا کرد و مشخص کرد هر پیکسل نسبت به همسایه های خودش روشن تره یا نه؟ برای هر پیکسل محاسباتش زیاد میشه و بهینه نیست. در نتیجه برای هر پیکسل میانگین همسایه هایش را حساب کرده و سپس مقدار پیکسل را با میانگین مقایسه میکنند.

	عملکرد	سرعت	نقطه ضعف	نقطه قوت

Otsu	محاسبه بر اساس مشخصه های آماری کل تصویر با کمک هیستوگرام آن	سرعت بالایی دارد. چون همان اول هیستوگرام تصویر مشخص میشود و بعد فقط باید برای ترشلد های مختلف ویژگی های آماری هر گروه را حساب کند و در فرمول بگذارد.	ویژگی های محلی را در نظر نمیگیرد و بخشی از داده ها گاهی از بین میرود بخاطر این ویژگی.	سرعت بالا. مبتنی بر ویژگی های تصویر.
adaptive threshold	محاسبه بر اساس مشخصه های آماری همسایه های هر پیکسل به طور جداگانه	سرعت بستگی به همسایگی ای که انتخاب میشود و تعداد پیکسل ها دارد. Otsu چون کانوولوشن هم سریع محاسبه میشود میتواند خوب باشد اما در کل از Otsu کند تر است چون برای هر پیکسل جدا حساب میشود.	سخت بودن پیدا کردن سطح آستانه. ویژگی های کل تصویر را در نظر نمیگیرد که گاهی خوب نیست. برای تصاویر ۳ و ۴ کاناله نیست.	برای تصاویری که نورپردازی مناسب نیست و اجزای تصویر قایل جداکردن با یک سطح آستانه نیستند بهتر است. مبتنی بر ویژگی های محلی.

سوال سوم) ب: مراحل اجرای الگوریتم adaptive threshold و توضیح پارامتر های cv2.adaptiveThreshold:

به جای اجرای استو در هر پیکسل => میانگین همسایه ها ی یک پیکسل حساب میشه. اگه روشن تر بود روشن و اگر تیره تر بود تیره در نظر میگیریم.

میشه با مرز پیدا کردن ناحیه رو پیدا کرد ولی یه جاهاییش اشتباه میشه.

الگوریتم:

برای هر پیکسل:

میانگین پیکسل ها اطراف را حساب کنید با یک کانولوشن ساده.

اگر پیکسل بزرگتر از میانگین شد: ۱

اگر کوچکتر بود: ۰

نتیجه رو با اصل مقایسه میکنیم و ذخیره میکنیم حاصل رو.

(Imag >= combinedimage) + C

فیلتر ۳ در ۳ خیلی کوچکه چون همسایه های نزدیک بهش رنگشون نزدیک به خودشه.

میانگین گیری میتونه وزن دار باشه مثلا گوسی باشه. به پیکسل های مرکزی وزن بیشتری بده هرچی دور میشیم وزنش کمتر بشه.

اولین کار کامباین کردن یا استفاده از فیلتر tobe.

پاسخ تمرین هشتم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

ناحیه بندی رو میشه با لبه یابی اصلاحش کرد.

adaptiveTreshold:

```
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
```

Src = یک تصویر به عنوان ورودی میگیره که باید تک کاناله باشه یعنی طیف خاکستری.

MaxValue : برای تعداد کلاس ها هست مثلا برای باینری یک میشه. ماکزیمم مقداریه که قراره به پیکسل های یک ناحیه متصل dst اطلاق بشه.

adaptiveMethod میتونه تابع میانگین یا گوسی باشه. اگه متود میانگین باشه.

میانگین: این تابع میانگین همسایه های blocksize*blocksize برای هر پیکسل رو حساب میکنه و منهای C میکنه.

گوسی: مجموعه همسایه ها رو به صورت وزن دار حساب میکنه و C رو ازش کم میکنه.

blockSize: سایز همسایه هایی که میخواد حساب کنه که میتونه مقادیر ۳و ۵و ۷و ... داشته باشه. به این معنی که برای محاسبه چه ماتریسی از همسایه ها باید محاسبه شود مثلا ۲۱ در ۲۱ یا ۳۳ در ۳۳. هرچه قدر بزرگ تر باشه به سمت global threshold میره هر چه قدر کوچکتر باشه نویزی میشه. پیکسل اگه بزرگتر از میانگین ساده یا وزندار اطرافش بود روشنه و اگه کوچکتر بود تیره است.

C: برای اینکه نویزی نشه مثلا در حالتی که میانگین ۸۰ باشه و اعداد ۷۹ و ۸۱ و این حدود ها باشن جواب یه سسری ۰ و ۱ میشه ک هیعنی نویز و ما میخوایم ناحیه ی نزدیک به هم همشون یه مقدار بشن. برای همین میایم میگیم مثلا ۵ واحد بیشتر از میانگین بود بشه ۱. برای اینکه ناحیه های یک دست همش ۰ یا سفید بشه میدیم میدیم مثبت ۵ و اگه بخوایم همش ۱ یا سیاه بشن میدیم منفی ۵. C رو معمولا صفر نمیدیم ولی میشه صفر هم داد. میگیم شرط روشن بودن اینه که از یه حدی روشن تر باشه و یا حد تیره بودن اینه که از یه حدی تیره تر باشه. مقدارش به نوع تابع اداپتیو هم بستگی داره.

Threshold type: دو نوع Thresh_binary و Thresh_binary_inv

- Here is the formula for `cv2.THRESH_BINARY`:

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxValue} \; if \; \texttt{src}\,(x,y) \; > \; \texttt{T(x,y)} \\ 0 \; otherwise \end{cases}$$

- Here is the formula for `cv2.THRESH_BINARY_INV`:

$$\texttt{dst}(x,y) = \begin{cases} 0 \; if \; \texttt{src}\,(x,y) \; > \; \texttt{T(x,y)} \\ \texttt{maxValue} \; otherwise \end{cases}$$

تصویر برداشته شده از کتاب: Mastering OpenCV4 with Python (Alberto Fernández Villán)

پاسخ تمرین هشتم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

سوال چهارم) پردازش متن از تصویر:

منبع: https://docs.opencv.org/3.4/d4/d76/tutorial_js_morphological_ops.html

https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html

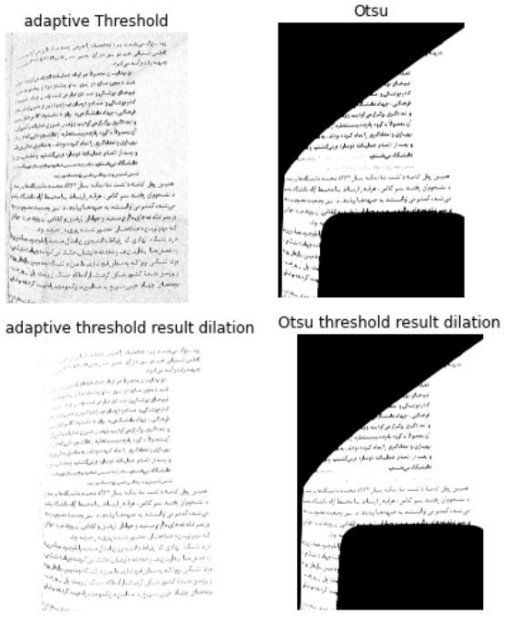
الف) پیش پردازش و پردازش روی تصویر royan.jpg :

برای adaptive threshold از تابع گوسی استفاده کردم که وزن نقاط نزدیک به پیکسل موردنظر در میانگین بیشتر باشد.

کرنل تولید شده با cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)) :

```
[[1 1 1]
 [1 1 1]
 [1 1 1]]
```

نتایج Otsu و adaptive threshold روی تصویر:



پاسخ تمرین هشتم

مهدیه نادری: ۹۸۵۲۲۰۷۶

نتایج متن تصویر اصلی:

تیم‌های بر

کادر پزشکی و ام
[فرهنگی «جهاد دا : ؟
و امدادگری برگزار می‌کردیم و ۳

آن معمولا با گروه یانزده‌بیست‌نفره؟

بهیاری و امدادگری را تمام کرده بودند. به

و بعد از اتمام عملیات ۳ برمی‌گشتیم ۱۹
دانشگاه می شد یم د:

۹ ٫ ۹

ال و3

سا که ۹ ۵ه یآ

همین روال ادامه داشتت ,تغا اینگ ه سال۱۳۶۲ محدد دانشگاه‌ها بازشد؛
دلنشی‌جویان» فخیی سر کلاس. هرقدر ارتباط با محیط ارام دانشگاه یتن
ی هب ری یج ومج او بیایند. با این وضعیت ۳۳

نان برد یی آن روزها بیاید جونای

.یه بود
اباوجود همه دوک
و «جهاد دانشگاهیا
نف های رو

ه آن روز ۳

کردند! مو |

هه ۳

نتایج متن Otsu:

ـ
ل

تیم‌های بر

کادر پزشکی و اه

فرهنگی , جیباد داز: 5

و امدادگری برگزار می‌کردیم و 5

آن معمولا با گروه یانزده‌بیست‌نفره ۹

! بهیاری و امدادگری را تمام کرده بودند. به

و بعد از اتمام عملیات دوباره برمیگشتیم ۲

دانشگاه می‌شدیم ۱ ار
همین روال ادامه داشت تااینکه سال ۱۳۶۲ محدد دانشگاه‌ها بازنده
دانشجویان رفتند سر کلاس. هرقدر ارتباط با محیط آرام دانشگاه بتن
می شد: مشرمی توانستند به جبیه ها بیایند. با این وضعیت محطوه بت
اش انقلابی آن روزها بیاید: حو-ی

4 نود .

.
تک ید

یاه حود همه دورد |

مجهاد دا ۱
دارای ۸ ود : ۱ ۳

اول آن روزف ب ۱
اک دند؛ موس

۳۵
ال
۰ ٫۷

۹
اره (ه
ریعنظ

نتایج متن adaptive threshold:

```
text on adaptive result image
زود بسزرگی می‌شدند و رام چندساله را عسر ۱۰ٔ ۳
تیا مد سم ۸ یا ام 9 رد ۳
نت ۰ و ی ۰

کاظمی اشتیان ی دسم در ایسن دو رن : سیم سا[ ۳۹ ۱۲

ا یاه عزام‌مایسان معصولا او ایام ۳ ۳ انا

7
ی ۳ د۵

تحدآن 0 روضان دو این ایام ب هرد ستتو 4 7 ۲
ات 1 ی

یر یر 2

تیم هاأی یزشتی 5 امدادی ی تیاز هی دس م4 ۰ تس
ما ی ی

_ کاذر یز کی و امد مداد و درمسان اد


سا ی ی ی
ا ی
:

فرهنگی ) جرهاد ۵ آنتر‌گاهی» ۲ برای دآفتسجود) اد 5 7 9
- خ ر ۱ ۰ ۱ ۹(« نی
و امدادگری برگزار می‌کردیم و زمان شروع عملیات رای

آن معمولا با گروه پانزدهبیست‌نفره از دانتسجودایی که
5 دهیاری و امدادتری و اتمام کرده پوندند. به مناطق اس جنگی مر
و بعد از اتمام عملیات دوباره برمی‌گتتیم ومشت ر
دانتسگاه می تند تیور (دکتر عبدالحسین شاهوردی. از دومتان دوران دانن
.(کاظم‌یأتیانی و مدیرعاصل فعاسی یژوهتگاه رویان
همین روال ادامه داتست تا اینکه سال ۱۳۶۲ محدد دانتسگاه‌ها بازتند

دانتسجویان رفتند سر کلاس . هرقدر ارتساط با محیط ارام دانتسگاه بیشتر
- می‌تد. کمترمی‌توانستند به جبهه ها پیایند. با این وضعیت معلوم نود
برسردغدغه‌های نا رام سعید وجوانان پرتور و انقلابی آن روز ها بای وی ۱
بهه بود.
ز را باه حود همه و
کرد «جیاانتن
نی زهای رز

بود؛تتسکلی نوا که به‌ستقلور یماد رتباط بره وا
: ار ماه سس یت بتن , ولویست اول آن ر ۳

ات
پیدا کردند؛ ما

مس ۵ میم درسن دعل تمتسان سای ره ار 2 ۲ 5

در دانتسگاه ال > تسا که | ارتباص دانتس‌جویان امتا لم فدسبکیت 1

انفصال‌هاء با آرمان‌ضا و دغدغه‌هایتان حفظ م کرد
دا ما وتأمین

ت
را ۳ زمانیو کتسور تتا , درا و سس

متر آ ۱ 1 1 ۱ ۱

بر یل ار ایا ۳ ۱
```

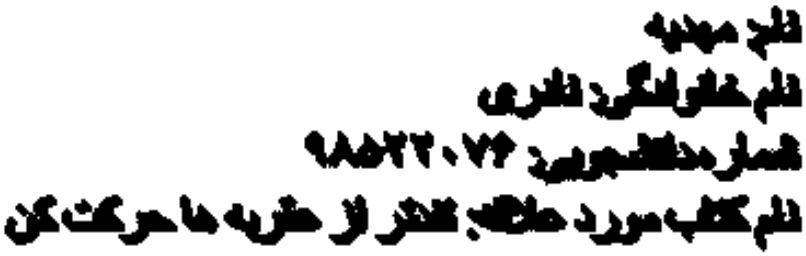
همانطور که مشخص است با مقایسه ی نوشته ها متوجه میشویم روش Otsu توانسته نوشته های سیاه رنگ را با وضوح خیلی بالایی مشخص کند و تابع گفته شده کلمات و جملات بیشتری تشخیص داده. اما به هر حال چون بخشی از تصویر را که در سایه افتاده بود کاملا سیاه کرده این بخش از متن کتاب در جملات آورده نشده و این خوب نیست.

در عوض الگوریتم آستانه گذاری وفقی یک مقدار تصویر تار تری تولید کرده اما متن و جملات بیشتری تشخیص داده است. زیرا با آستانه گذاری محلی توانسته قسمت هایی از متن که در سایه بودند را هم تشخیص دهد و بیشتر کلمات و جملات را استخراج نماید.

پاسخ تمرین هشتم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

ب)پیش پردازش و پردازش روی تصویر mypic.png :

تصویر ساخته شده با فیلتر paint strokes:

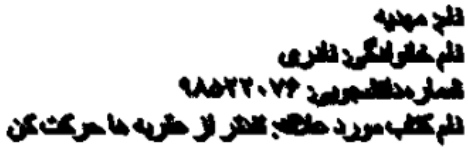


کرنل تولید شده با cv2.getStructuringElement(cv2.MORPH_RECT, (1,1)) :

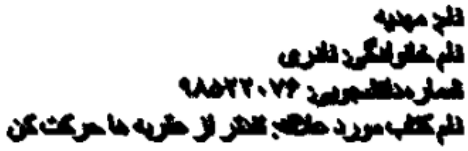
```
[[1 1]]
```

نتایج آستانه گذاری های مختلف:

Original Image

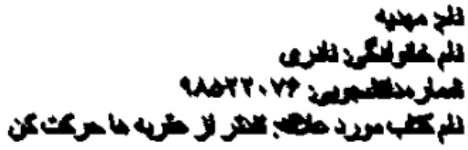


Otsu

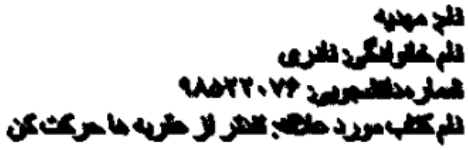


```
[[1 1]]
```

Otsu threshold result dilation

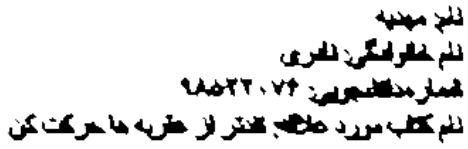


adaptive Threshold

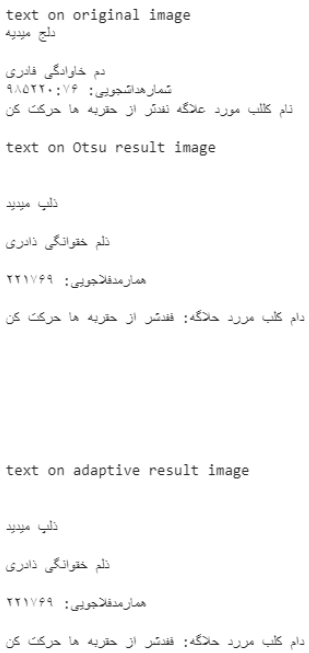


```
[[1 1]]
```

adaptive threshold result dilation



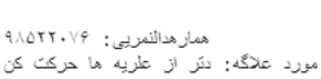
متن های بدست آمده با پردازش برای پیش پردازش های مختلف بالا:



نتیجه گیری:

در اینجا تصویر اصلی سایه و یا نورپردازی متفاوت در نقاط مختلف نداشت. خود تصویر هم وضوح خوبی داشت اما نوشته ها نا مشخص بود و کمی حروف کنار هم درهم فرورفته بودند. قطعه کد گفته شده را وقتی روی تصویر خام اجرا کردیم همه ی خطوط پیدا شدند اما حروف و شماره ها در بعضی کلمات اشتباه تشخیص داده شده بودند.

وقتی کرنل `[[1]\n[1]]` را برای الگوریتم Otsu استفاده کردم نتوانست خط مربوط به نام خانوادگی را درست تشخیص دهد. همچنین حروف کلمات شماره دانشجویی را هم اشتباه تشخیص داد اما اعداد را درست متوجه شد. برای همین از کرنل `[[1 1]]` برای هر دو الگوریتم استفاده کردم ولی اعداد را این کرنل کاملا اشتباه نشان میدهد.



این کرنل ۲ در ۱ جزئیات افقی را حذف میکرد و کمک میکرد اعداد از هم جدا شده تشخیص داده شوند. کرنل ۱ در ۲ سطر بیشتری را تشخیص میداد.

در نهایت جواب هر دو کرنل با یک کرنل ۱ در ۲ مشابه هم دیگر شد.