گزارش تمرین هشتم بینایی ماشین

نام تهیه‌کننده: ملیکا نوبختیان

شماره دانشجویی: ۹۷۵۲۲۰۹۴

نسخه: ۱

۱- سوال اول

در ابتدا یک توضیح کلی در مورد الگوریتم splitting and merging و مراحل آن ارائه می‌دهیم.

مراحل این الگوریتم به صورت زیر است:

۱. ابتدا ناحیه کلی segmentation، معیار homogeneity و ساختمان داده هرمی خود را تعریف می‌کنیم.

۲. اگر هر ناحیه R در این ساختار هرمی homogenous نبود، این ناحیه را به ۴ بخش کوچک‌تر تقسیم می‌کنیم. اگر هر کدام از این ناحیه‌ها با parent یکسان این قابلیت را داشتند که با هم merge شوند و به یک ناحیه homogenous تبدیل شوند این کار را انجام می‌دهیم. اگر هیچ ناحیه‌ای باقی نمانده بود که split یا merge شود به مرحله بعد می‌رویم.

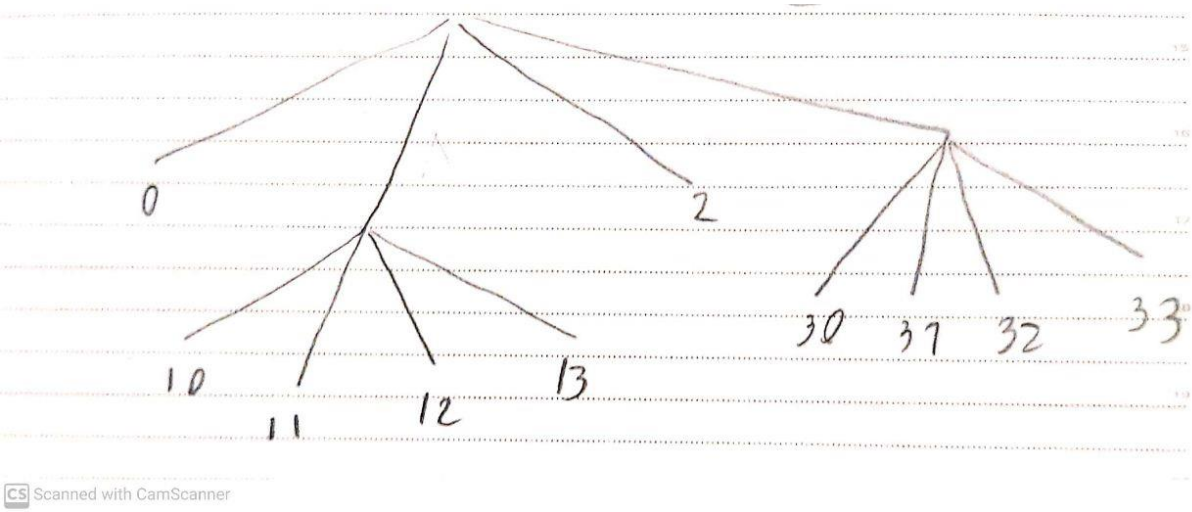
۳. اگر دو ناحیه مجاور با merge شدن می‌توانند یک ناحیه homogenous تشکیل دهند این کار را انجام می‌دهیم هر چند در سطوح مختلفی از ساختار هرمی باشند و یا parent یکسان نداشته باشند.

حالا سعی می‌کنیم این مراحل را روی نمونه داده شده انجام دهیم. در ابتدا عملیات splitting را انجام می‌دهیم و تا زمانی که تمام نواحی که به دست می‌آوریم homogenous باشند:

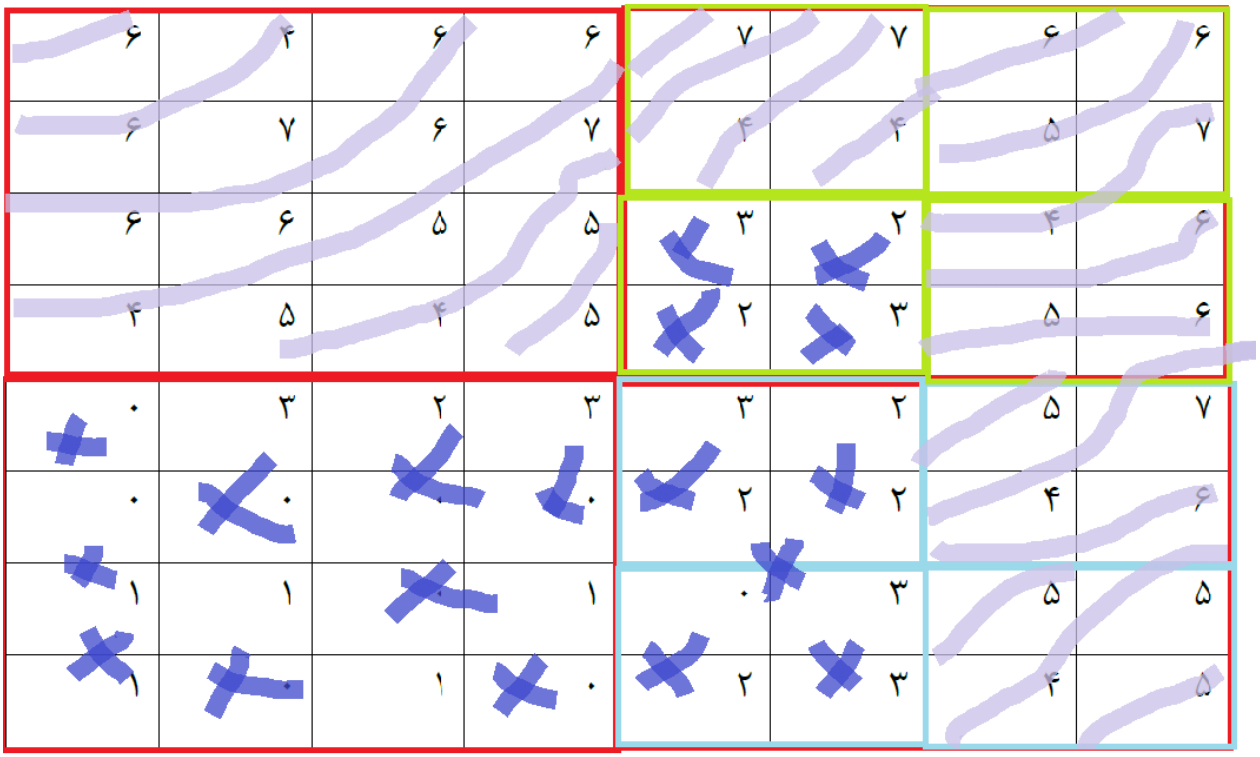
۶	۶	۷	۷	۶	۴	۶	۶
۷	۵	۴	۴	۶	۷	۶	۷
۶	۴	۳	۲	۵	۶	۵	۵
۶	۵	۲	۳	۴	۵	۴	۵
۷	۵	۳	۲	۲	۳	۲	۳
۶	۴	۲	۲	۰	۰	۰	۰
۵	۵	۰	۳	۱	۱	۰	۱
۵	۴	۲	۳	۰	۰	۱	۱

در ابتدا تصویر را به چهار ناحیه ۴*۴ تقسیم می‌کنیم. ناحیه صفر (ناحیه بالا سمت چپ) homogenous است زیرا تفاوت بیشترین و کمترین مقدار در آن ۳ است که شرط ما هم این است که اگر میزان اختلاف در ناحیه‌ای کمتر مساوی ۳ بود آن ناحیه homogenous است. در مورد ناحیه ۲ (پایین سمت چپ) نیز همین حالت را

داریم. اما نواحی ۱ و ۳ homogenous نیستند زیرا در ناحیه ۱ اختلاف ۵ و در ناحیه ۳ اختلاف برابر ۷ است پس این نواحی باید دوباره split شوند. پس از اینکه دوباره split کردیم دیگر ناحیه‌ای وجود ندارد که homogenous نباشد. ساختار هرمی ما به شکل زیر خواهد بود:



در ابتدا سعی می‌کنیم نواحی که دارای parent یکسان و در یک سطح را با هم merge کنیم. سپس به سراغ بقیه نواحی می‌رویم. در نهایت به شکل زیر خواهیم رسید:



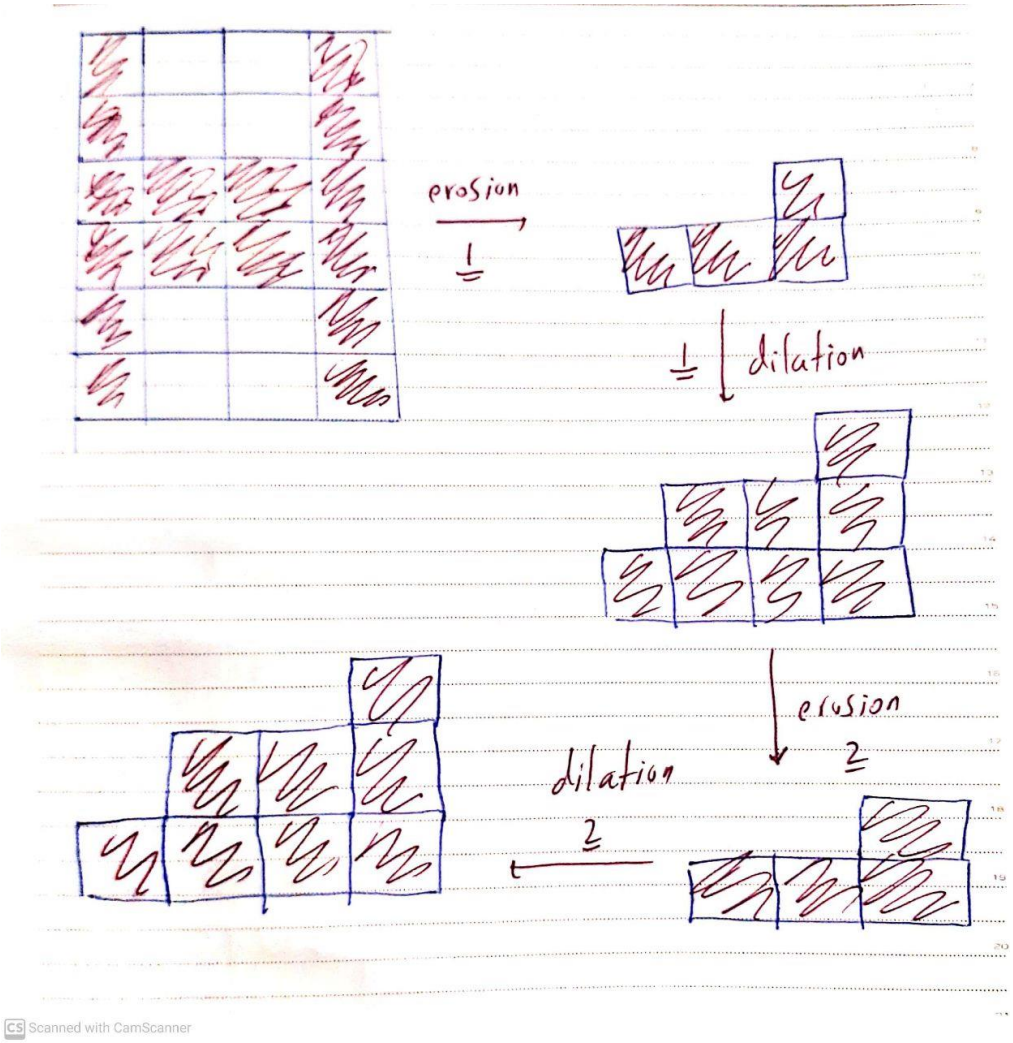
ب) در region growing ما یک روش bottom-up را در پیش می‌گیریم. به این شکل که ابتدا یک پیکسل را به عنوان seed در نظر می‌گیریم و با در نظر گرفتن یک threshold و بررسی پیکسل‌های همسایه، پیکسل‌های واجد شرایط را به آن پیکسل اضافه می‌کنیم. به این ترتیب به تدریج ناحیه‌ها شکل می‌گیرند.

در region splitting and merging ما به شکل عکس حالت قبل عمل می‌کنیم. به این شکل که ابتدا یک تصویر را یک ناحیه در نظر می‌گیریم و به تدریج و در صورتی که آن ناحیه شرط threshold را رعایت نکند به چهار قسمت کوچک‌تر تقسیم خواهد شد تا در نهایت همه ناحیه‌ها واجد شرایط شوند. سپس نواحی مجاوری که در صورت ترکیب شدن باز هم واجد شرایط خواهند بود ترکیب خواهند شد. می‌توان گفت این روش به نحوی top-down است.

۲- سوال دوم

اجرای عملگر باز برای یک بار روی تصویر تاثیر خود را بر تصویر می‌گذارد و نواحی که باید را حذف و نواحی دیگر را تقویت می‌کند. اما اجرای این عملگر برای بار دوم هیچ تاثیری نخواهد داشت. زیرا نواحی که درابتدا توسط erosion حذف خواهند شد، دوباره توسط dilation اضافه خواهند شد. به عبارت دیگر با اجرای یک بار عملگر باز روی یک تصویر، یک عنصر خنثی نسبت به عملگر باز می‌سازیم.

نتیجه اجرای این عملگر روی شکل سوال به این صورت خواهد شد:



۳- سوال سوم

الف) از منظر سرعت otsu سرعت بیشتری خواهد داشت زیرا یک threshold کلی را برای کل تصویر در نظر می‌گیرد و تنها بین ۲۵۶ مقدار باید یک عدد را انتخاب کند. اما در adaptive این کار محلی است و بسته به همسایگی برای هر پیکسل باید جداگانه این کار را انجام دهیم که این عملیات را زمان بر می‌کند.

از دیدگاه عملکرد اگر تصویری که داریم سایه در قسمت‌های مختلف آن متفاوت باشد، otsu عملکرد خوبی نخواهد داشت، زیرا به طور global عمل می‌کند و برای نواحی مختلف به شکل جداگانه عمل نمی‌کند اما adaptive در اینجا بهتر عمل می‌کند چون برای هر قسمت threshold متفاوتی در نظر می‌گیرد.

ب) روش کار به این صورت است که برای هر پیکسل متناسب با همسایگی که برای آن در نظر گرفتیم یک آستانه مناسب پیدا می‌کنیم و تا در نهایت به مقدار مناسب برای هر پیکسل برسیم.

در مورد تابع، ورودی اول تصویر grayscale است. ورودی دوم maxvalue است که بر اساس روش thresholding مقادیر کوچک‌تر یا بزرگ‌تر از T به maxvalue نگاشت خواهند شد. پارامتر سوم روشی که برای adaptive thersholding استفاده می‌کنیم و یک مقدار مناسب آستانه را به دست می‌آوریم. یکی از این روش‌ها روش گاوسی است که از میانگین گاوسی در همسایگی استفاده می‌کند. پارامتر بعدی روش thresholding را مشخص می‌کند. یکی از این روش‌ها THRESH_BINARY است. پارامتر بعدی neighbourhood size که اندازه همسایگی برای یک پیکسل را مشخص می‌کند. در آخر نیز پارامتر بعدی ثابتی را مشخص می‌کند که از میانگین کم خواهد شد.

۴- سوال چهارم

در گام اول تصویر موردنظر را می‌خوانیم و آن را تبدیل به grayscale می‌کنیم:

```
img = cv2.imread("royan.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

برای اعمال otsu از تابع cv2.threshold به شکل زیر استفاده می‌کنیم:

```
(T, threshInv) = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRES
H_OTSU)
```

ورودی اولی که به این تابع می‌دهیم تصویر grayscale است. ورودی دوم threshold را مشخص می‌کند که در اینجا چون otsu را داریم بی معنا است و مقدار صفر به آن داده‌ایم. ورودی سوم maxvalue ما را مشخص می‌کند که بر اساس روش thresholding مقادیر کوچک‌تر یا بزرگ‌تر از T به maxvalue نگاشت خواهند شد. در اینجا چون از THRESH_BINARY استفاده کرده‌ایم مقادیر بزرگ‌تر از T به maxvalue نگاشت خواهند شد. flag دیگری که داریم THRESH_OTSU است که otsu را اجرا خواهد کرد.

خروجی‌ها نیز threshInv است که تصویری است که الگوریتم روی آن اجرا شده‌است و T در نهایت Threshold بهینه‌ای است که توسط otsu انتخاب شده‌است.

در مرحله بعد سعی می‌کنیم یک عنصر ساختاری مناسب تعریف کنیم و عملیات dilation را انجام دهیم. برای این کار از یک عنصر ساختاری ۳*۳ به شکل RECT استفاده می‌کنیم:

```
1 cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
```

```
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 1]], dtype=uint8)
```

این عملگر را به تابع dilate می‌دهیم تا عملیات dilation را انجام دهد. تعداد iteration موردنیاز را هم برابر ۲ قرار می‌دهیم:

```
kernelSize = (3, 3)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, kernelSize)
iteration = 1
dilation = cv2.dilate(threshInv.copy(), kernel, iterations= iteration)
```

در نهایت متنی که با این روش استخراج می‌شود به شکل زیر خواهد بود:

```
۲
تیم‌های بو

کادر پزشکی وا

فرهنگی «جهاد دانش_ کل

و آمادگری برگزار می‌کردیم و ظ

آن معمولا با کروه باتزده‌بیست نفره

بهباری و امدادگری را تمام کرده بودند. به ۴

و بعد از اتمام عملیات دوباره برمی‌کشتیم و۲
دانشگاه مبی شسد یم ی در

همین روال ادامه داشت تااینکه سال ۱۳۶۲ مجدد دانشگاه‌ه با شده
دانشجویان رفتند سر کلاس. هرقدر ارتباط با محیط ارام دانسگاه بشن
می شد #مترمی‌توانستند به جبهه‌ها پیاین ,با این وضعیت معنوه بت
شب وانفلابی ان رء وزها پیید؛ حو-ی

۰3۰
نف : _ ص
```

در قسمت بعد به جای otsu از adaptive threshold استفاده می‌کنیم:

```
thresh = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 21, 3)
```

ورودی اول تصویر grayscale است. ورودی دوم همان maxvalue در حالت قبل است. پارامتر سوم روشی که برای adaptive thersholding استفاده می‌کنیم. در اینجا ما از روش گاوسی استفاده می‌کنیم که از میانگین گاوسی در همسایگی استفاده می‌کند. پارامتر بعدی روش thresholding را مشخص می‌کند. در اینجا هم مانند حالت قبل از THRESH_BINARY استفاده کردیم. پارامتر بعدی neighbourhood size که اندازه همسایگی را مشخص می‌کند که در اینجا ۲۱*۲۱ است. در آخر نیز پارامتر بعدی ثابتی را مشخص می‌کند که از میانگین کم خواهد شد.

نتیجه اجرا این حالت نیز به شکل زیر است:

```
مِ مه

رو ت سزر ۵ می دنل و ر آم س خر له ۱ أ تس ی و ۴
ی ۳ شیم ۹ دشسم ۵ ۹ این ك ور ۰ ۳۹ إ با ۱ ۳۳ : ۱ ۵ " ۱
ی " زا با بش ۳
ی رشت وأمن دم کت ۵ ۳

هم ۳۹۹ چم ۹ ۴ ۲ |
| اک امد ۱۳ تصبا ۹ ۷ و ایا مر تیاه

3
تعدآد مجروصان ۹ ایس ایام ستو یو و ۱
تیم ضای یدنب سخی ۹ امه بادی از هبی ی ت ۳
کادر پزشکی و امداد و درمسان نیساز بین. ۴
| نی دی
فرهنکی «حهاد دانشگاهی». برای « دجم هد ۳ »۳
و امد ادگری برکز ار می ترذ یم و زمسان لتسون سای ۱ بِ ۱
آن معم ما یا گروه یانزده بیست نفره از دای تا یب
بهباری ۵ امدآدگری را تمام کرده نت یه ماش ی ۰
ِ و بعل ۹ ز اتمام عملیات دوباره برمی‌کش: سیم و دس ۲

دانشگاه می دیجم (نکششر عبدالحسسین شساهورد تا و ندال
.(کاظلمی‌آشتیانی و مدیرعاسل فعلی پژوهشگاهد رویان
همیسن رول ادامه داشت تا اینکه سال ۱۳۶۲ محدد دانشگاه‌ها بازشد؛
می‌شد . کمترمی‌توانستند به جبهه‌ها بیایند. با این وضعیت مطوم ی
بر سر دغدغه‌های ناارام سید و حوانان پرشورو انقلابی آن روزها بباید؛ جوانلیر
. بر در یره بود ۹۳
```

۲

۳

رای ؟ ۱ ِ شلی سای أ
۳ " « در ی مس لب شیاه ولـه مت و اوعان از ار سوم
ت ی کر بخ رد

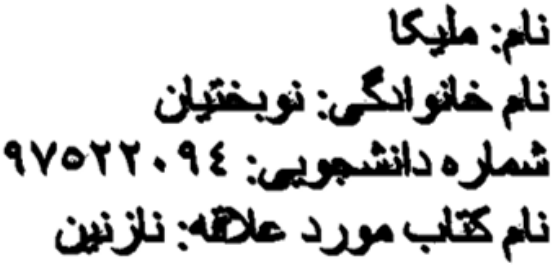
2 2 4 ۱ ۱ ی مر

2 ۲ ۳ ۲ ـِـ ۱ تست ول »۳ و ۱ ۱ ِ ۵ ۱ ۳ ی ۳ هم و () ی شیی د
7 که ۹ ژ ۰ ۰ ۳ ۱ کر
لل" موس
۵ ۱ أ ستَ ۱ و ۱ ۱ ه ۱ اِ حدم
ر گـر زان بـل دام ۶ ۰ ۱ ژِ ۲ ۱ ۳ ۱ سس دی ۳۳ وید ۹ - ی ی بت
۱ سس ۱ ریـبـد ر ا

در این حالت چون از adaptive استفاده کرده‌ایم توانسته‌ایم متن بیشتری را بازیابی کنیم. زیرا تصویری که داریم در نواحی مختلف سایه‌های متفاوتی دارد و در این صورت اجرای otsu نتیجه خیلی خوبی در بر نخواهد داشت. زیرا otsu تنها یک threshold استفاده می‌کند که در مورد ما مناسب نیست.

ب)

در قسمت دوم تصویر ورودی من به شکل زیر خواهد بود:



خروجی کد زیر را نیز مشاهده می‌کنید:

```
img = cv2.imread("mypic.png")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
(T, threshInv) = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRES
H_OTSU)
kernelSize = (3, 3)
kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, kernelSize)
iteration = 2
dilation = cv2.dilate(threshInv.copy(), kernel, iterations= iteration)
```

```
#################################################################
# Finding contours
im2 = img.copy()
contours, hierarchy = cv2.findContours(dilation, cv2.RETR_EXTERNAL,cv2.CHA
IN_APPROX_NONE)

for cnt in contours:
  x, y, w, h = cv2.boundingRect(cnt)
  # Drawing a rectangle on copied image
  rect = cv2.rectangle(im2, (x, y), (x + w, y + h), (0, 255, 0), 2)
  # Cropping the text block for giving input to OCR
  cropped = im2[y:y + h, x:x + w]
  print(pytesseract.image_to_string(cropped, lang='fas'))
```

نام: ملیکا

نام خانوادگی: نوبختیان

شماره دانشجویی: ۷۵۲۲۰۹۶۹
نام کتاب مورد علاقه: نازنین