Computer vision

Homework 8

Parsa Eissazadeh 97412364

سوال 1

در ابتدا نیاز داریم یک معیار برای تعیین همگونی تعریف کنیم . این معیار را من این گونه تعریف میکنم که حداکثر اختلاف بین سطح روشنایی پیکسل ها 3 باشد .
در مرحله اول عکس را به چهار قسمت تقسیم میکنیم و اینکار را آنقدر ادامه میدهیم تا 4 بخش یک عکس (یا یک زیر عکس) همگن باشند .

یک چهارم های عکس :

بالا چپ :

6	4	6	6
6	7	6	7
6	6	5	5
4	5	4	5

حداکثر اختلاف 3 است پس همگن است و نیازی به split کردن دوباره نیست .

پایین چپ :

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

حداکثر اختلاف 3 است و نیازی به split نیست .

پایین راست :

3	2	5	7

2	2	4	6
0	3	5	5
2	3	4	5

هم 7 داریم هم 2 پس نیازی به split نیست .

بالا راست :

7	7	6	6
4	4	5	7
3	2	4	6
2	3	5	6

هم دو داریم هم 7 پس همگن نیست .

جداول سمت راست را باید split کنیم :

جدول سمت راست پایین :

a1	a2
a3	a4

جدول سمت بالا :

b1	b2
b3	b4

جدول a1 :

2	3
2	2

جدول a2 :

7	5
6	4

جدول a3 :

3	0
3	2

جدول a4 :

5	5
5	4

جدول b1 :

7	7
4	4

جدول b2 :

6	6
7	5

جدول b3 :

2	3
3	2

جدول b4 :

6	4
6	5

حال همه ی عکس ها همگن هستند .

حال برای شباهت دو زیرعکس مشابه یک معیار تعریف می کنیم آن هم اینکه حداکثر اختلاف بین دو پیکسل همچنان کمتر از 3 باشد . حال عکس هایی را که بازه مشترک دارند با هم ادغام میکنیم . حتی آنهایی که **اندازه** و **والد** یکسان ندارند .

در ابتدا که عکس های پایین چپ و بالا چپ قطعا همگن نیستند در نتیجه دو رنگ متفاوت دارند :

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

سپس عکس های a1 و a3 نیز همه بین 0 تا 3 هستند و همه شبیه یک چهارم پایین سمت چپ هستند :

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

				0	0	0	0
				0	0	0	0

b2 ، B1 و b4 هر سه در یک بازه اند و مجاور هم اند :

I	I	I	I	I	I	I	I
I	I	I	I	I	I	I	I
I	I	I	I			I	I
I	I	I	I			I	I
				0	0	0	0
				0	0	0	0
				0	0	0	0
				0	0	0	0

A2 و a4 در یک بازه اند و بازه شان با b4 مشترک است که مجاور آن هستند . همچنین b3 که مجاور a1 است بازه ای مشترک با آن دارد . در نتیجه عکس نهایی به شکل زیر است :

I	I	I	I	I	I	I	I
I	I	I	I	I	I	I	I
I	I	I	I	0	0	I	I
I	I	I	I	0	0	I	I
0	0	0	0	0	0	I	I
0	0	0	0	0	0	I	I
0	0	0	0	0	0	I	I
0	0	0	0	0	0	I	I

ب)

در الگوریتم split & merge ابتدا عکس ها به زیر عکس هایی که تقسیم می شوند که هر کدامشان همگن هستند (با هر معیاری) و اندازه های مختلفی دارند و سپس زیرعکس های مشابه با هم ادغام می شوند .

مرحله دوم این الگوریتم (ادغام شدن) با region growing شباهت دارد ، اما تفاوتی که دارد این است که در هر بررسی ، فقط یک پیکسل به ناحیه اضافه نمی شود و کل segment ای که در آن قرار دارد به آن ملحق می شود که سرعت الگوریتم را افزایش می دهد .

سوال 2

در عملگر باز در ابتدا یک سایش میزنیم (برای حذف نویز ها) سپس یک گسترش.

- از آنجایی که عکس های باینری هستند ، برای نمایش عکس ها از یک جدول استفاده میکنم که مقادیر باینری دارند . یک padding با مقدار 0 هم اضافه میکنیم . (یعنی ابعاد جدول 6 در 8 خواهد بود)

در سایش ،جاهایی که structuring element ما زیر مجموعه عکس است 1 میگذاریم در غیر این صورت صفر .

بار اول

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

سپس گسترش می زنیم . در افزایش هر جا که structuring element ما اشتراکی با عکس داشت 1 میگذاریم:

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

بار دوم

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

گسترش :

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	1	1

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

برای راستی آزمایی کد در نوتبوک هم پیاده سازی شده است .

سوال 3

الگوریتم otsu :

نقاط ضعف : اگر مثلا در عکس سایه وجود داشته باشد و یک نقطه بسیار تاریک باشد و نقطه دیگر خیلی روشن (به عبارتی هیستوگرام خیلی equalize شده باشد و در همه رنگ ها به تعداد یکسان پیکسل داشته باشیم) ، در یک بخشی از تصویر ناحیه های روشن اصلا دیده نمی شوند و در بخش دیگر تصویر ناحیه های تاریک اصلا دیده نمی شود . و این مشکل با هیچ threshold ای حل نمی شود چون حد آستانه ها برای همه پیکسل های عکس یکی است .

نقاط قوت : خیلی ساده (با زمان و پیچیدگی کم) میتواند پیکسل ها را به دو دسته سیاه و سفید تقسیم کند .

الگوریتم adaptive threshold :

نقاط ضعف : با اینکه الگوریتم میانگین یا میانه روی تعداد محدودی از پیکسل های مجاور میگیرد و ساده تر است ولی از آنجاییکه این الگوریتم را برای همه پیکسل ها اعمال میکند ، ممکن است عکسی بسیار بزرگ باشد و این الگوریتم زمان زیادی بگیرد .

نقاط قوت : چون برای هر پیکسل یک threshold جدید حساب میکند ، کیفیت تصویر بسیار بالاتر است و مشکل otsu را حل میکند .

مراحل اجرای الگوریتم adaptive threshold :

- یک فیلتر میانگین گیر تعریف میکنیم . این فیلتر آرایه ای است که همه 1 اند و اگر اندازه ش w*w باشد باید بر یک w^2 تقسیم بشود .
- این فیلتر میانگین گیر را در عکس کانوالو میکنیم . (همیشه با کانوالو کردن یک اضافه کردن padding هم داریم) (این فیلتر می تواند هر چیزی باشد که صرفا اطلاعاتی از پیکسل های مجاور به ما بدهد ، مثلا یک فیلتر گاوسی هم میتواند باشد)
- نتیجه این میانگین گرفتن را از یک عدد ثابتی کم میکنیم . (c)
- حال میخواهیم اختلاف سطح روشنایی هر پیکسل با خانه های مجاورش را حساب کنیم ، پس عکس را منهای این نتیجه کانوالو میکنیم .
- اگر این اختلاف از حدی (یک threshold) بیشتر بود ، آن را روشن و در غیر این صورت تاریک میگذاریمش .

پارامتر های متد `adaptiveThreshold` :

لینک استفاده شده :

https://www.tutorialspoint.com/opencv/opencv_adaptive_threshold.htm#:~:text=Adaptive%20thresholding%20is%20the%20method,()%20of%20the%20Imgproc%20class.

پارامتر ها :

- Src : عکسی که پردازش های رویش انجام می شوند .
- maxValue : بعد از دو سطحی کردن ، به پیکسل ها یکی از دو عدد 0 و maxValue را مقدار میدهیم .
- adaptiveMethod : فیلتری که در عکس ورودی کانوالوش میکنیم . (که در پیاده سازی openCV یا میانگین میگیرد یا فیلتر گاوسی اعمال میکند)
- Threshold type : کلا دو حالت دارد . بعد از دو سطحی کردن مشخص می کند که خانه های بالاتر از حد میانگین ، مقدارشان 0 باشد یا آن maxValue باشد .
- Block size : انداز فیلتر

- C : همان عدد ثابت

dst - An object of the class Mat representing the destination (output) image.

maxValue - A variable of double type representing the value that is to be given if pixel value is more than the threshold value.

thresholdType - A variable of integer type representing the type of threshold to be used.

سوال 4

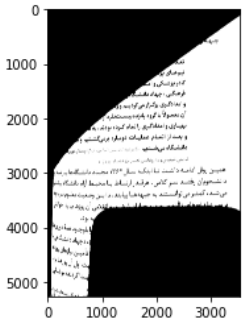
الف)

به کمک متد cvtColor عکس را به gray تبدیل کردیم .

سپس باید هم otsu را اعمال می کردیم هم adaptive thresholding را .

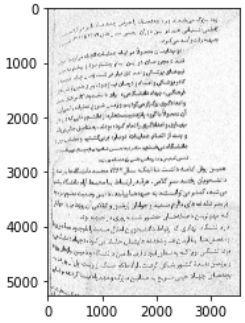
```
[35] # applying otsu
     ret,otsu_thresh = cv2.threshold(gray,127,200,cv2.THRESH_BINARY)

     plt.imshow(otsu_thresh,cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7f03d341d8e0>
```




```
[36] # applying adaptive otsu
     ada_otsu_thresh = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 11, 2)

     plt.imshow(ada_otsu_thresh, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f03d33e6af0>
```



استفاده از متد های threshold و adaptive threshold :

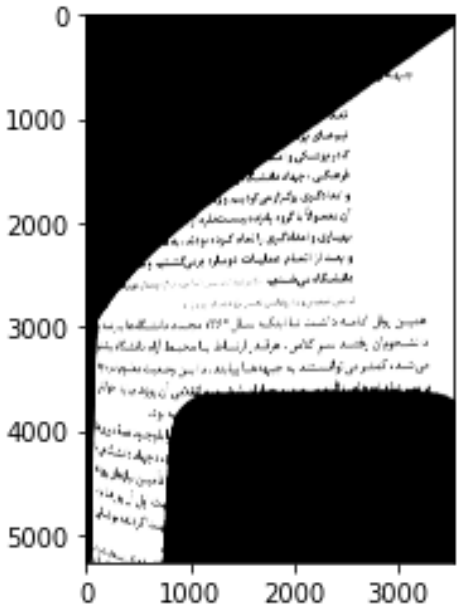
https://gist.github.com/pknowledge/6fd9944c2345b8872b8917257f88a7d1

سپس تابع use_pytesseract را تعریف میکنیم :

```
# using pytesseract
def use_pytesseract (img_path):
  extractedInformation = pytesseract.image_to_string(img_path, lang='fas')
  return extractedInformation
```

سپس خروجی هر دو متد را به آن پاس می دهیم :

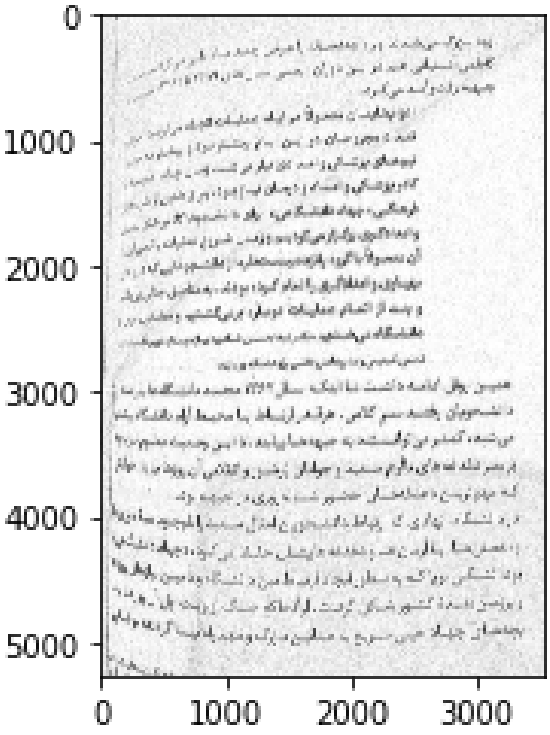
خروجی الگوریتم Otsu :



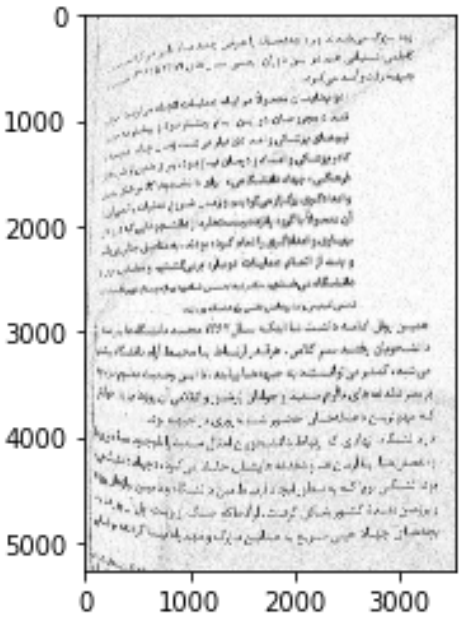
برای خروجی الگوریتم Otsu ، خروجی بسیار بهتر از حالت عادی عکس بود و متن های بیشتری پیدا شدند :

```
نیم‌های پ

کادر پزشکی و ام

[ فرهنگی «جیاد داز

و امدادگری برگزار می‌کردیم و ۳

آن معمولا با گروه پانزده‌بیست‌نفره ۲۳

بمهیاری و امدادگری ی را تمام کرده بودند. به۲

و بعد از اتمام عملیات دوباره بمکشتیم و

دانشگاه می شدیم .- سید ریت .از دوستان دوران
و تاسلمی‌اشتیاقی و مدیرعامل فعلی بزوهش رک :بویا

همین روال ادامه دانست تااینکه ث"لت محدد دانشگاه‌ها بازشده

دانشجویان رفتند سر کلاس. هرقدر ارتباط با محیط ارام دانشگاه بشت

می‌شد. کمترمی‌توانستند به جبهه‌ها بیایند. با این وضعیت معلوم نبود جه

ق بی آن روزها بیاید؛ ؛ حواانی ۳ ۳

ض ۱
"یار جود همه دورک
«جهاد دا
اتأمسن نبازهای روز
تال تا
اکردند؛ م9

۱
۳

ره
رده راد
```

اما در خروجی Otsu همچنان بخش زیادی از متن سیاه است در نتیجه معلوم نیست .
در خروجی الگوریتم adaptive threshold ، متد py_tesseract تقریبا خروجی ای نداد .
خروجی الگوریتم :

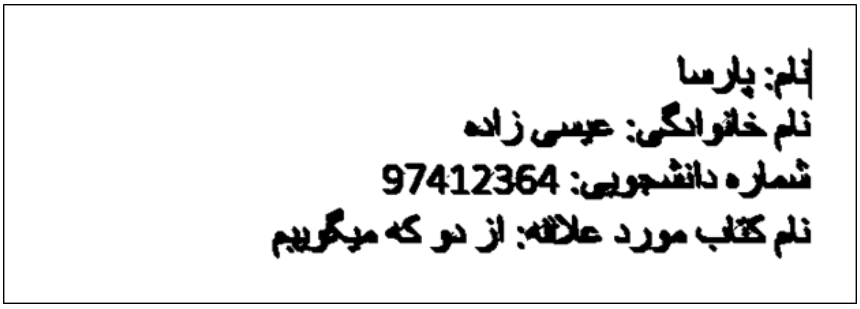


حدس من این بود که مشکل از نازک بودن فونت ها باشد برای همین یک عملگر باز بر روی آن اعمال کردم.



ب)

در ابتدا نویز خواسته شده را روی عکس اعمال میکنیم . عکس به شکل زیر شد :



سپس از متد getStructuringElement استفاده میکنیم تا پنجره های مختلف از قبل تعریف شده openCV را پیدا کنیم :

```
# Structuring element
KERNEL_LENGTH = 3
KERNEL_SIZE = (KERNEL_LENGTH,KERNEL_LENGTH)

rect = cv2.getStructuringElement ( cv2.MORPH_RECT , KERNEL_SIZE)
ellipse = cv2.getStructuringElement ( cv2.MORPH_ELLIPSE , KERNEL_SIZE)
cross = cv2.getStructuringElement ( cv2.MORPH_CROSS , KERNEL_SIZE)

structuring_elements = [
    rect ,
    ellipse ,
    cross
]

structuring_elements_names = [
    "rect" ,
    "ellipse" ,
    "cross"
]
```

استفاده از متدهای getStructuringElement و dilate :

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

سپس آن ها را درون آرایه ای میریزیم و یکی یکی به عکس اعمال کردیم و در subplot ها نمایششان دادیم :

برای اندازه پنجره 3 :

برای اندازه پنجره 5 :

هر چه اندازه پنجره بزرگتر باشد نوشته ها بیشتر از بین میروند . زیرا هر چه بزرگتر بشوند احتمال اینکه زیر مجموعه عکس باشند کمتر می شود . در فیلتر اندازه 9 عکس به کلی از بین میرود .

تقریبا ellipse با اندازه 3 از بقیه بهتر است و همان را اعمال میکنیم .

```
[114] dilation = cv2.dilate(otsu_thresh , structuring_elements[1] , iterations = 1)
```

سپس هر دو را به کدی که قبلا در سوال نوشته شده بود پاس میدهیم .