به نام خدا

درس مبانی بینایی کامپیوتر

تمرین سری هشتم

مدرس درس:
جناب آقای دکتر محمدی

تهیه شده توسط:
الناز رضایی ۹۸۴۱۱۳۸۷

تاریخ ارسال: ۱۴۰۱/۰۹/۱۳

سوال ۱ :

با خواندن قسمت ۶.۳.۳ کتاب بارگذاری شده به سوالات زیر پاسخ دهید (۳۰ نمره)

- الف) سیگنال دو بعدی زیر را به روش Splitting and merging، با حد آستانه "۳" تقسیم بندی کنید. (مرحله به مرحله این کار را بنویسید)

6	4	6	6	7	7	6	6
6	7	6	7	4	4	5	7
6	6	5	5	3	2	4	6
4	5	4	5	2	3	5	6
0	3	2	3	3	2	5	7
0	0	0	0	2	2	4	6
1	1	0	1	0	3	5	5
1	0	1	0	2	3	4	5

- ب) تفاوت و شباهت‌های بین رویکرد Splitting and merging و Region growing ،را بیان نمایید . (هر کدام از الگوریتم‌ها را نیز توضیح دهید)

پاسخ ۱:

الف) این روش به این صورت است که ابتدا تا جایی که شرطمان satisfy تشده است، عمل splitting را در ناحیه‌ای که شرط satisfy نیست، انجام دهیم. سپس نواحی را با هم مقایسه کرده و در صورتی که شرطمان برای merge کردن دو ناحیه satisfy بود، آن‌ها را merge می‌کنیم. شرطی که برای split کردن نواحی استفاده می‌کنیم، این است که اختلاف بیشینه و کمینه مقدار پیکسل‌های یک ناحیه، کوچک‌تر یا مساوی حد آستانه باشد. همچنین شرطی که برای merge کردن دو ناحیه استفاده می‌کنیم، این است که اختلاف ماکزیمم ناحیه اول و مینیمم ناحیه دوم و همچنین اختلاف ماکزیمم ناحیه دوم و مینیمم ناحیه اول از حد آستانه کمتر یا مساوی آن باشد. حال شروع به حل این سوال می‌کنیم.

$step1$:

$max = 7, min = 0 \Rightarrow |max - min| = 7 > 3 \Rightarrow splitting\checkmark$

6	4	6	6	7	7	6	6
6	7	6	7	4	4	5	7
6	6	5	5	3	2	4	6
4	5	4	5	2	3	5	6
0	3	2	3	3	2	5	7
0	0	0	0	2	2	4	6
1	1	0	1	0	3	5	5
1	0	1	0	2	3	4	5

R1 R2 R3 R4

شکل حاصل پس از مرحله اول

$step2$:

حال شرط splitting را برای ناحیه‌های ۱ تا ۴ بررسی می‌کنیم.

$R1 : max = 7, min = 4 \Rightarrow |max - min| = 3 \leq 3 \Rightarrow splitting\times$

$R2 : max = 3, min = 0 \Rightarrow |max - min| = 3 \leq 3 \Rightarrow splitting\times$

$R3 : max = 7, min = 2 \Rightarrow |max - min| = 5 > 3 \Rightarrow splitting\checkmark$

$R4 : max = 7, min = 0 \Rightarrow |max - min| = 7 > 3 \Rightarrow splitting\checkmark$

6	4	6	6	7	7	6	6
6	7	6	7	4	4	5	7
6	6	5	5	3	2	4	6
4	5	4	5	2	3	5	6
0	3	2	3	3	2	5	7
0	0	0	0	2	2	4	6
1	1	0	1	0	3	5	5
1	0	1	0	2	3	4	5

R1 R2 R3 R4 R5 R6 R7 R8 R9 R10

شکل حاصل پس از مرحله دوم

$step3$:

در ادامه، شرط splitting را برای ناحیه‌های ۳ تا ۱۰ بررسی می‌کنیم.

$$R3 : max = 7, min = 4 \Rightarrow |max - min| = 3 \leq 3 \Rightarrow splitting\times$$

$$R4 : max = 7, min = 5 \Rightarrow |max - min| = 2 \leq 3 \Rightarrow splitting\times$$

$$R5 : max = 3, min = 2 \Rightarrow |max - min| = 1 \leq 3 \Rightarrow splitting\times$$

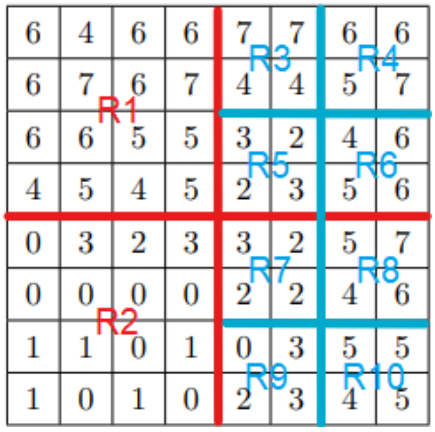
$$R6 : max = 6, min = 4 \Rightarrow |max - min| = 2 \leq 3 \Rightarrow splitting\times$$

$$R7 : max = 3, min = 2 \Rightarrow |max - min| = 1 \leq 3 \Rightarrow splitting\times$$

$$R8 : max = 7, min = 4 \Rightarrow |max - min| = 3 \leq 3 \Rightarrow splitting\times$$

$$R9 : max = 3, min = 0 \Rightarrow |max - min| = 3 \leq 3 \Rightarrow splitting\times$$

$$R10 : max = 5, min = 4 \Rightarrow |max - min| = 1 \leq 3 \Rightarrow splitting\times$$



شکل حاصل پس از مرحله سوم

$step4$:

با پایان مرحله splitting، شروع به بررسی شرایط merge کردن، و در نهایت merging ناحیه‌ها می‌پردازیم.

$$\left.\begin{array}{l} |max(R1) - min(R2)| = 7\times \\ |max(R2) - min(R1)| = 3\checkmark \end{array}\right\} R1\ and\ R2\ can\ not\ be\ merged$$

$$\left.\begin{array}{l} |max(R1) - min(R3)| = 3\checkmark \\ |max(R3) - min(R1)| = 3\checkmark \end{array}\right\} R1\ and\ R3\ can\ be\ merged$$

$$
\left.\begin{array}{l} |max(R1) - min(R5)| = 5\times \\ |max(R5) - min(R1)| = 1\checkmark \end{array}\right\} R1\ and\ R5\ can\ not\ be\ merged
$$

$$
\left.\begin{array}{l} |max(R4) - min(R3)| = 3\checkmark \\ |max(R3) - min(R4)| = 2\checkmark \end{array}\right\} R3\ and\ R4\ can\ be\ merged
$$

$$
\left.\begin{array}{l} |max(R6) - min(R5)| = 4\times \\ |max(R5) - min(R6)| = 1\checkmark \end{array}\right\} R5\ and\ R6\ can\ not\ be\ merged
$$

$$
\left.\begin{array}{l} |max(R5) - min(R3)| = 1\checkmark \\ |max(R3) - min(R5)| = 5\times \end{array}\right\} R3\ and\ R5\ can\ not\ be\ merged
$$

$$
\left.\begin{array}{l} |max(R4) - min(R6)| = 3\checkmark \\ |max(R6) - min(R4)| = 1\checkmark \end{array}\right\} R4\ and\ R6\ can\ be\ merged
$$

$$
\left.\begin{array}{l} |max(R5) - min(R7)| = 1\checkmark \\ |max(R7) - min(R5)| = 1\checkmark \end{array}\right\} R5\ and\ R7\ can\ be\ merged
$$

$$
\left.\begin{array}{l} |max(R8) - min(R6)| = 3\checkmark \\ |max(R6) - min(R8)| = 2\checkmark \end{array}\right\} R6\ and\ R8\ can\ be\ merged
$$

$$
\left.\begin{array}{l} |max(R2) - min(R7)| = 1\checkmark \\ |max(R7) - min(R2)| = 3\checkmark \end{array}\right\} R2\ and\ R7\ can\ be\ merged
$$

$$
\left.\begin{array}{l} |max(R2) - min(R9)| = 3\checkmark \\ |max(R9) - min(R2)| = 3\checkmark \end{array}\right\} R2\ and\ R9\ can\ be\ merged
$$

$$
\left.\begin{array}{l} |max(R7) - min(R8)| = 1\checkmark \\ |max(R8) - min(R7)| = 5\times \end{array}\right\} R7\ and\ R8\ can\ not\ be\ merged
$$

$$
\left.\begin{array}{l} |max(R9) - min(R10)| = 1\checkmark \\ |max(R10) - min(R9)| = 5\times \end{array}\right\} R9\ and\ R10\ can\ not\ be\ merged
$$

$$
\left.\begin{array}{l} |max(R7) - min(R9)| = 3\checkmark \\ |max(R9) - min(R7)| = 1\checkmark \end{array}\right\} R7\ and\ R9\ can\ be\ merged
$$

$$
\left.\begin{array}{l} |max(R8) - min(R10)| = 3\checkmark \\ |max(R10) - min(R8)| = 1\checkmark \end{array}\right\} R8\ and\ R10\ can\ be\ merged
$$

بنابراین، با توجه به نتایج به دست آمده از بخش بالا، سیگنال ما به شکل زیر در می‌آید:

6	4	6	6	7	7	6	6
6	7	6	7	4	4	5	7
6	6	5	5	3	2	4	6
4	5	4	5	2	3	5	6
0	3	2	3	3	2	5	7
0	0	0	0	2	2	4	6
1	1	0	1	0	3	5	5
1	0	1	0	2	3	4	5

شکل نهایی

ب) در روش Region growing، ابتدا یک پیکسل را به عنوان پیکسل بذر (seed) در نظر گرفته و آن را با پیکسل‌های مجاور خودش مقایسه می‌کنیم. اگر پیکسل‌های مجاور از قوانین از پیش تعریف شده پیروی کنند (مثلا اختلاف آن‌ها از یک حدی با پیکسل بذر بیشتر نباشد)، آن پیکسل به ناحیه پیکسل بذر اضافه می‌شود و این روند تا جایی ادامه پیدا می‌کند که هیچ شباهتی باقی نماند.

در مورد روش splitting and merging، ابتدا تصویر را به صورت یک منطقه واحد در نظر می‌گیریم. اگر منطقه ما، از قوانین تعریف شده پیروی نکرد (مانند قوانین توضیح داده شده در قسمت الف)، آن را به چندین ناحیه (معمولا ۴) تقسیم می‌کنیم و این روند ادامه پیدا می‌کند تا جایی که همه ناحیه‌ها از قوانین پیروی کنند. پس از پایان این مرحله (splitting)، به سراغ گام بعد (merging) می‌رویم. در این تکنیک، ما هر پیکسل را به عنوان یک منطقه مجزا در نظر می‌گیریم. به این صورت که یک منطقه را به عنوان منطقه بذر انتخاب می‌کنیم تا بررسی کنیم که آیا مناطق مجاور بر اساس قوانین از پیش تعریف شده مشابه هستند یا خیر. اگر مشابه باشند، آن‌ها را در یک منطقه mergeمی‌کنیم و به منظور ساختن مناطق segment شده کل تصویر به جلو حرکت می‌کنیم. هر دو فرآیند region splitting و region merging فرآیندهای تکراری هستند. معمولاً ابتدا region splitting بر روی یک تصویر انجام می شود تا یک تصویر به حداکثر مناطق تقسیم شود و سپس این مناطق merge می‌شوند تا تصویر قطعه‌بندی شده خوبی از تصویر اصلی ایجاد شود. برای region splitting می‌توان شرایط زیر را بررسی نمود تا در مورد تقسیم یا عدم تقسیم یک ناحیه تصمیم گرفت. اگر مقدار مطلق اختلاف

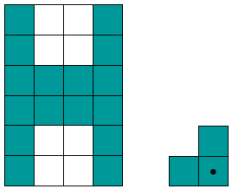
حداکثر و حداقل شدت پیکسل در یک منطقه کمتر یا مساوی با مقدار آستانه‌ای باشد که کاربر تعیین می‌کند، آن ناحیه نیازی به splitting بیشتر ندارد.

در بالا، به طور مختصر به شباهت‌ها و تفاوت‌های دو روش گفته شده، اشاره شد. اما به طور کلی تکراری بودن جز شباهت‌ها، و از تفاوت‌ها می‌توان این مورد را لحاظ کرد که در region growing یک پیکسل را با پیکسل بذر مقایسه می‌کنیم، اما در region splitting and region merging، برای splitting مقدار اختلاف ماکزیمم و مینیمم یک ناحیه با حد آستانه سنجیده می‌شود و برای merging، ناحیه‌ها را با هم مقایسه می‌کنیم. به طور کلی، هدف هر دو الگوریتم، ناحیه‌بندی یک تصویر می‌باشد.

منبع:

https://towardsdatascience.com/image-segmentation-part-2-8959b609d268

سوال ۲:

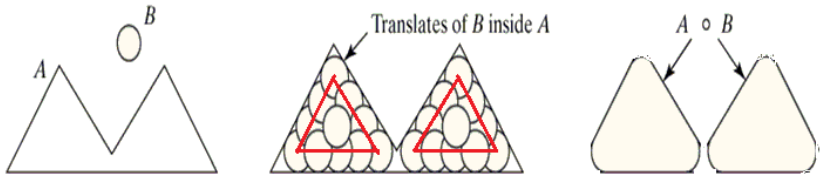


تفاوت بین اجرای عملگر باز را برای ۱ یا ۲ بار بنویسید. سپس، نتیجه ۲ بار اجرای این عملگر را با توجه عنصر ساختاری و تصویر داده شده، ترسیم کنید. (۱۵ نمره)

پاسخ ۲:

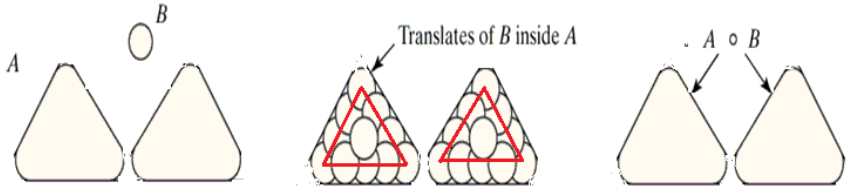
در عملگر باز، ابتدا عملگر سایش، و سپس عملگر گسترش روی تصویر اعمال می‌شود. کاربرد این عملگر، برای حذف جزئیات کوچک و هموار کردن محیط نواحی تعریف شده می‌باشد. همچنین تفاوتی برای اجرای ۱ یا n بار اجرای این عملگر وجود ندارد. برای اثبات این موضوع، تصویر زیر را در نظر بگیرید. در این تصویر، تصویر اولیه و عنصر ساختاری و همچنین نتیجه یک بار عملگر باز بر روی تصویر آورده شده است.

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

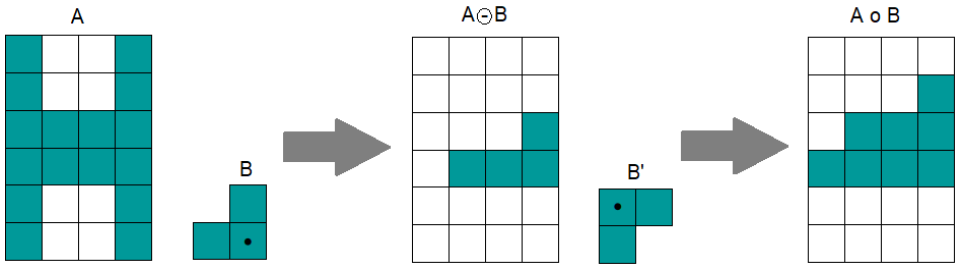


نتیجه این عملگر پس از ۲ بار، مطابق شکل زیر می‌شود.

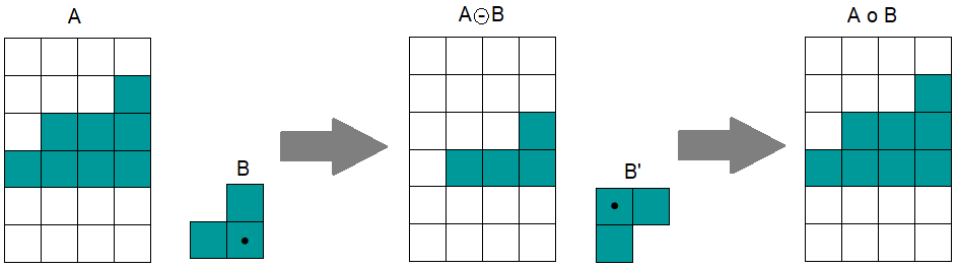
$$A \circ B = (A \ominus B) \oplus B$$



همانطور که دیده می‌شود، تفاوتی در تصویر حاصل از اعمال این عملگر پس از ۱ و ۲ بار وجود ندارد. حال، این عملگر را با عنصر ساختاری داده شده، بر روی تصویر داده شده اعمال می‌کنیم. طبق توضیحات داده شده در بالا، تصویر حاصل پس از یک بار اعمال عملگر باز، معادل اعمال کردن ۲ بار آن است. با این حال، من دو بار این عملگر را اعمال کردم تا دوباره این موضوع ثابت شود.



نتیجه پس از اعمال یک بار عملگر باز



نتیجه پس از اعمال دو بار عملگر باز

سوال ۳:

به سواالت زیر پاسخ دهید. (۵ نمره)

- الف) نقاط ضعف و قوت ۲ الگوریتم otsu و adaptive threshold را بیان کنید. (حتما از منظر سرعت و عملکرد دو الگوریتم را قیاس نمایید.)
- ب) مراحل اجرای الگوریتم adaptive threshold را دقیقا توضیح داده و پارامترهای تابع cv2.adaptiveThreshold را شرح دهید.

پاسخ ۳:

- الف) الگوریتم otsu روشی برای تعیین سطح آستانه براساس مشخصه‌های آماری می‌باشد. به طور خلاصه این الگوریتم، سطح آستانه‌ای را انتخاب می‌کند که با انتخاب آن، رنگ‌های هر گروه مشابه باشند (واریانس بین پیکسل‌های هر کلاس کمینه شود). از مشکلات این روش، می‌توان به درست کار نکردن در بعضی حالات، مثلا زمانی که در تصویر سایه داریم، یا در بخشی از تصویر تجمع پیکسل‌های سیاه زیاد است و در بخشی تجمع پیکسل‌های سفید خیلی زیاد است، هنگامی که تصویر نویزی باشد، زمانی که ناحیه شی در مقایسه با ناحیه پس زمینه کوچک باشد، اشاره کرد. همچنین سرعت بالا، یکی از مزایای این روش است؛ چرا که برای یک تصویر ۸ بیتی، سطح آستانه یکی از مقادیر ۰ تا ۲۵۵ است.

  برای حل مشکلات روش قبل، adaptive thereshold معرفی شد. ایده این روش این بود که برای جاهایی که تصویر روشن‌تر است، از سطح آستانه بالاتر، و جاهایی که تصویر

تیره‌تر است، از سطح آستانه کوچک‌تری استفاده کنیم. اگر تصویر را مثلا به ۴ ناحیه تقسیم می‌کردیم، باز هم ممکن بود باعث مشکل شود و اختلاف بسیاری بین نواحی نواحی رخ می‌داد و تصویر ناهمگنی حاصل می‌شد؛ بنابراین پیشنهاد شد حد آستانه برای هر پیکسل محاسبه شود، اما این راه هم از لحاظ محاسباتی پیچیده می‌شد و هزینه زیادی به همراه داشت. راه حل ارائه شده برای این مشکل، استفاده از پیکسل‌های همسایه یک ناحیه و تقریب حد آستانه با استفاده از میانگین یا میانه آن‌ها می‌باشد که در روش adaptive thereshold، از این راه استقاده می‌کنیم. اشکال این روش این است که محاسباتی گران است و بنابراین برای کاربردهای بلادرنگ مناسب نیست. اندازه محله باید به اندازه کافی بزرگ باشد تا پیکسل‌های پیش‌زمینه و پس‌زمینه کافی را پوشش دهد، در غیر این صورت یک آستانه ضعیف انتخاب می‌شود. همچنین از مزایای این روش، می‌توان به حل مشکلات otsu، و عملکرد بهتر آن در شرایط مختلف اشاره کرد.

به طور خلاصه اگر بخواهیم این دو روش رو با هم مقایسه کنیم، otsu از لحاظ سرعت سریع‌تر است و adaptive threshold عملکرد بهتری دارد.

- ب) در الگوریتم adaptive thereshold، ابتدا یک تصویر gray scale را به عنوان ورودی می‌گیریم. در ادامه با استفاده از میانگین یا میانه پیکسل‌های اطراف یک ناحیه کوچک، حد آستانه را برای آن ناحیه کوچک تخمین می‌زنیم. بنابراین ما حد آستانه‌ها مختلفی برای ناحیه‌های متفاوت داریم. سپس با استفاده از حد آستانه‌های به دست آمده برای هر ناحیه، تصویر را باینری می‌کنیم.

  در مورد پارامترهای این تابع، اولین پارامتر، src یا تصویر ورودی می‌باشد. دومین پارامتر (maxValue) است که عددی نامنفی است و به پیکسل‌هایی که از حد آستانه بیشتر هستند، اختصاص می‌یابد. سومین پارامتر، (adaptiveMethod) است که الگوریتم مورد استفاده برای adaptive theresholding را مشخص می‌کند که می‌تواند GAUSSIAN یا MEAN باشد. پارامتر بعدی، thresholdType است که نوع thereshold را مشخص می‌کند که باید THRESH_BINARY یا THRESH_BINARY_INV باشد. پنجمین پارامتر، blockSize است که سایز همسایگی پیکسل‌هایی که برای محاسبه حد آستانه استفاده می‌شوند را تعیین می‌کند. آخرین پارامتر نیز c می‌باشد که عدد ثابتی است که از میانگین یا میانگین

وزن‌دار کم می‌شود. خروجی این تابع نیز یک تصویر با همان ابعاد ورودی اما در حالت باینری می‌باشد. این تابع به همراه ورودی‌هایش در ادامه آورده شده است.

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

سوال ۴:

در این قسمت از تمرین می‌خواهیم با Optical character recognition ،یک تصویر را گرفته و سپس متن موجود در تصویر را چاپ کنیم که در آن از کتابخانه pytesseract استفاده می‌کنیم. اما با توجه به اینکه می‌خواهیم عبارات با زبان فارسی را بخوانیم، نیاز به یک‌سری پیش‌پردازش خواهیم داشت. پیش‌پردازش‌های زیر را برای دو قسمت تمرین در فایل HW[8].ipynb طی کنید و در هر بخش خروجی را نمایش دهید. (۵۰ نمره)

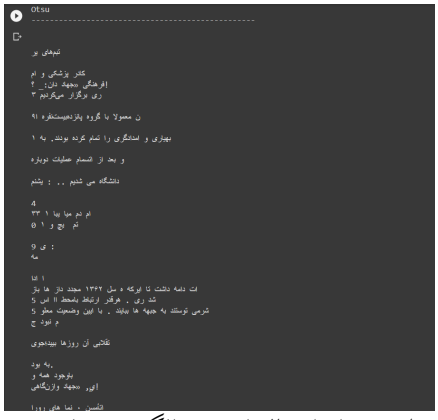
- تصویر را خوانده و با تابع cv2.cvtColor ،آن را به سطح خاکستری ببرید و در متغیر gray بریزید.
- با تابع cv2.threshold بر روی آن otsu رو پیاده کنید.
- سپس با تابع cv2.getStructuringElement یک عنصر ساختاری مناسب تعریف نموده و با تابع cv2.dilate روی خروجی otsu ، عملیات مورفولوژی را انجام دهید و خروجی را در متغیر dilation بریزید.
- بدون پیش‌پردازش‌های بالا، تصویر خام را به تابع pytesseract.string_to_image دهید و متن‌های خروجی را با هم مقایسه کنید.

۱. در قسمت اول این تمرین، تصویر royan.jpg را بخوانید، و صرفا مراحل i و ii را اجرا نمایید. اما در مرحله ii در کنار الگوریتم otsu، تابع cv2.adaptiveThreshold هم به کار ببرید و خروجی این دو تابع را به pytesseract.string_to_image بدهید و نتیجه را مقایسه نمایید.

۲. در قسمت دوم تمرین، فایل word داده شده را طبق شکل زیر، با اطلاعات خودتان تکمیل نموده، و به صورت تصویر ذخیره کرده و سپس در فایل word مجدد آپلود نموده و افکت Strokes paint را روی آن اجرا کنید. و در مرحله آخر به صورت تصویر ، به نام mypic.png ذخیره نمایید و سپس مراحل پیش‌پردازش فوق را اجرا نمایید.

پاسخ ۴:

۱. همانطور که در تصاویر زیر نیز مشخص است، هنگامی که از adaptive theresholding استفاده می‌کنیم، خروجی بهتر و واضح‌تری خواهیم داشت. در سوال قبل، به علت این موضوع اشاره شد.



برای نمایش بهتر این موضوع، من تصاویر حاصل پس از اعمال این دو الگوریتم را نیز در تصویر زیر نمایش دادم.

همانطور که در تصویر بالا نیز مشخص است، الگوریتم otsu به علت آستانه‌گذاری یک سطحی، در بخش‌هایی از تصویر که سایه داریم، عملکرد بسیار ضعیفی دارد؛ اما در روش آستانه‌گذاری وفقی، ما این مشکل را نداریم.

۲. نتیجه خروجی این بخش، به شرح زیر است:

همانطور که دیده می‌شود، متن word به طور کامل برای ما نمایش داده شده است؛ اما اگر بدون استفاده از این پیش‌پردازش‌ها بخواهیم از تابع pytesseract.image_to_string استفاده کنیم، هیچ چیزی به ما نمایش نمی‌دهد.