

## به نام خدا

استاد: دکتر محمدرضا محمدی  
درس مبانی بینایی کامپیوتر

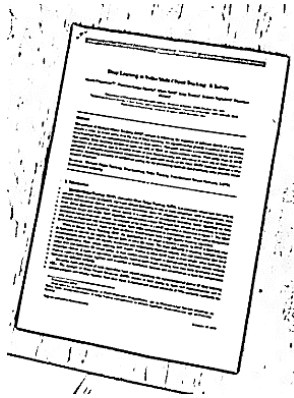
نام: فاطمه زهرا بخشنده  
شماره دانشجویی: 98522157

ابتدا تصویر را می‌خوانیم و نمایش می‌دهیم.

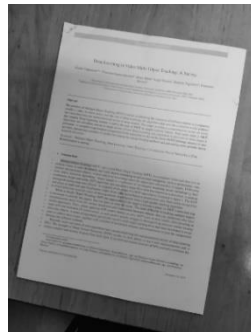
**1-** نگاشت سیاه و سفید: تصویر رنگی BGR را GrayScale کرده و در grayscale ذخیره می‌کنیم.



**2-** محو کردن تصویر: در این مرحله می‌خواهیم تصویر را برای مراحل بعدی به ویژه لبه یابی آماده کنیم. می‌توانیم از Gaussian Filter یا Bilateral Filter استفاده کنیم. در این مرحله می‌توانیم پس از استفاده از یک فیلتر گاوسی، از adaptiveThreshold که تازه در درس آموختیم نیز استفاده کرده و سپس روی آن fastNIMeansDenoising هم بزنیم تا تصویر باینری داشته باشیم و لبه یابی دقیقتر انجام شود. با این روش، در انتهای این مرحله تصویر زیر را خواهیم داشت.



فیلتر Bilateral Filter نسخه پیشرفته فیلتر گاوسی است که برای صاف کردن تصاویر و کاهش نویز و در عین حال حفظ لبه ها استفاده می شود. در رویکرد دوم می توانیم از این فیلتر استفاده کرده و تنها blurring انجام دهیم و از thresholding صرف نظر کنیم. در این صورت نتیجه به صورت زیر است:



3- تشخیص لبه: از لبه یاب Canny استفاده می کنیم. با توجه به اینکه از چه رویکردی در مرحله قبل استفاده می کنیم باید threshold ها را کمی متفاوت تنظیم کنیم. نتیجه با رویکرد اول:



نتیجه با رویکرد دوم:

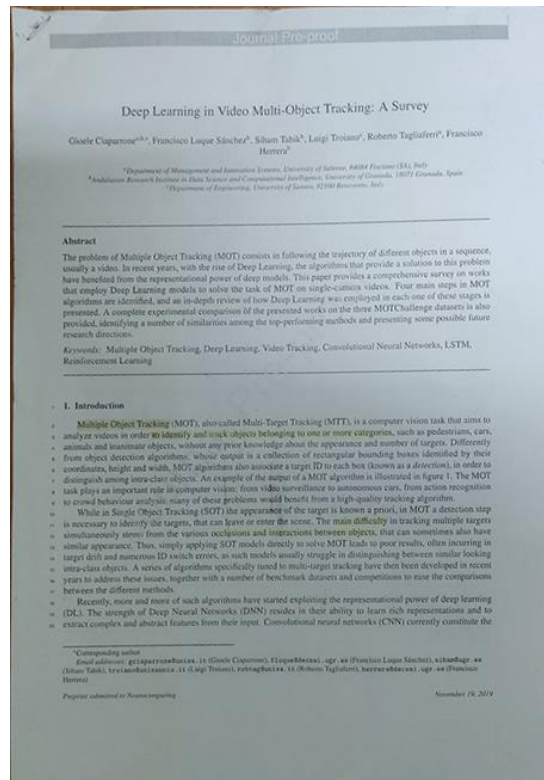


**4-** تشخیص رئوس: از تابع `findContours` در `OpenCV` استفاده می‌کنیم که تمام نقاط پیوسته (در امتداد لبه‌ها) که رنگ یا شدت یکسانی دارند را باهم `join` می‌کند. چون می‌خواهیم در آخر تنها 4 نقطه گوشه برگه را ذخیره کنیم، `Contour Approximation Method` را `CHAIN_APPROX_SIMPLE` قرار می‌دهیم.

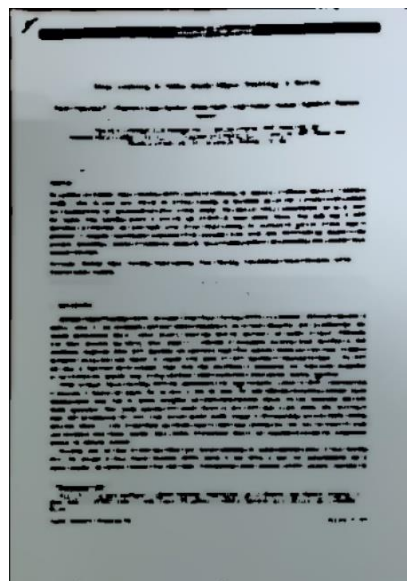
برای محاسبه بزرگترین `contour` در تصویر، روی `contour` های پیدا شده `loop` می‌زنیم و برای هر `contour`، از `arclength` و `approxPolyDP` استفاده می‌کنیم که بسته به دقتی که ما مشخص می‌کنیم، یک شکل کانتور را به شکل دیگری با تعداد رئوس کمتر تقریب می‌زند. اگر `len(approx) > 4` باشد، `isContourConvex` برای آن `True` باشد و `contourArea` برای آن از مساحت تقریبی خود تصویر کوچکتر، و از بزرگترین `area` ای که تا الان پیدا شده بزرگتر باشد، آنگاه `contour` را برابر این `contour` قرار می‌دهیم. در نهایت بزرگترین `contour` با همین روش پیدا می‌شود. نتیجه روی تصویر:



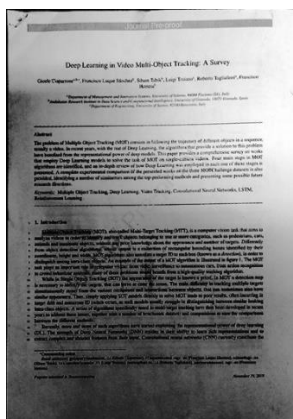
5- نگاشت دورنما و برش: ابتدا vertices را با استفاده از تابع `reorder`، به ترتیب بالا چپ، بالا راست، پایین راست و پایین چپ، مرتب می‌کنیم. `width` تصویر جدید را محاسبه می‌کنیم، که حداکثر فاصله بین مختصات  $x$  پایین راست و پایین چپ یا مختصات  $x$  بالا راست و بالا چپ است. همچنین ارتفاع تصویر جدید را محاسبه می‌کنیم، که حداکثر فاصله بین مختصات  $y$  بالا راست و پایین راست یا مختصات  $y$  بالا چپ و پایین چپ است. از تابع `getPerspectiveTransform` برای پیدا کردن تبدیلی استفاده می‌کنیم که منطقه‌ای را که شناسایی کرده‌ایم به یک مستطیل که به راحتی می‌توانیم آن را برش دهیم، `map` می‌کند. سپس باید از `cv2.warpPerspective` برای اعمال این تبدیل و برش منطقه هدف استفاده کنیم. نتیجه تصویر `crop` شده:



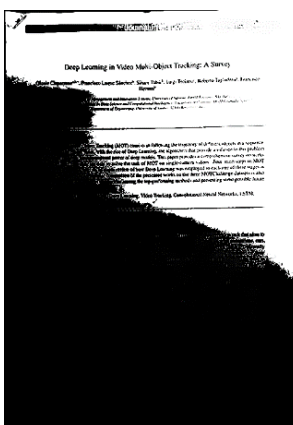
- 6- مرحله آخر، بهبود تصویر: برای بهبود تصویر کراپ شده، روش های زیادی را جستجو و امتحان کردم، که کد آن ها در انتهای نوت بوک موجود است. برخی از آن ها نتیجه قابل قبولی نداشتند. برای مثال: 1. روش cartooning: این روش برای این تصویر نتیجه خوبی نداشت اما اگر عکس در شرایط بهتری گرفته شود می تواند نتیجه زیبایی داشته باشد.



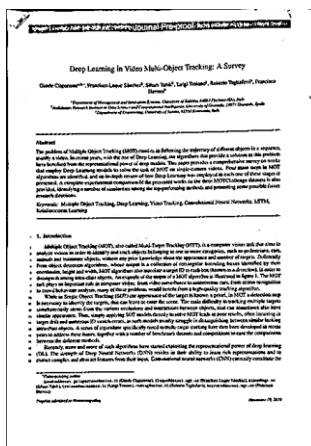
2. histogram equalization: نتیجه قابل قبول نبود.



3. thresholding: نتیجه برای این تصویر که نور آن بد بود اصلا قابل قبول نبود.

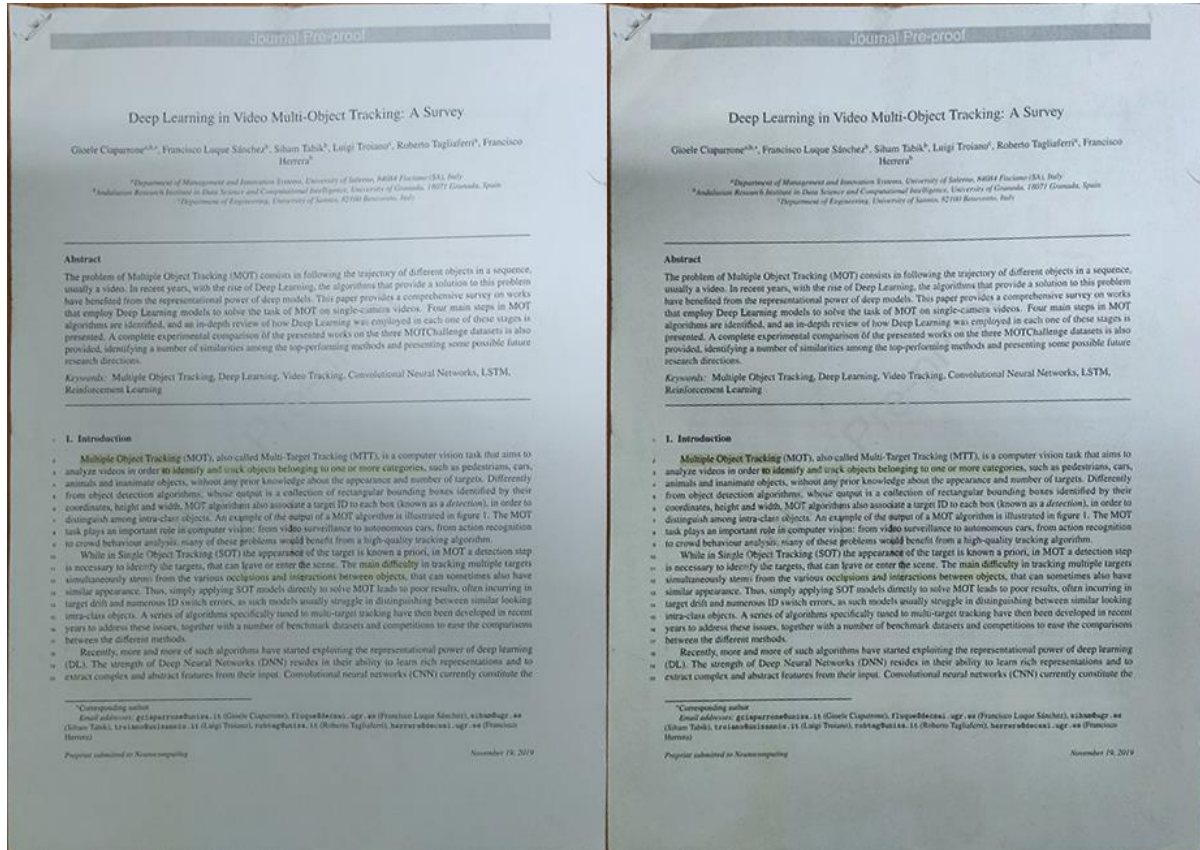


4. نتیجه adaptive thresholding: این روش از thresholding معمولی پیشرفته تر است و معمولا کار آمد است. اگر تصویر کمی در شرایط بهتری گرفته شود می تواند بسیار عالی عمل کند. در ادامه می بینیم برای تصویر دیگری که ورودی می دهیم خیلی خوب عمل می کند.



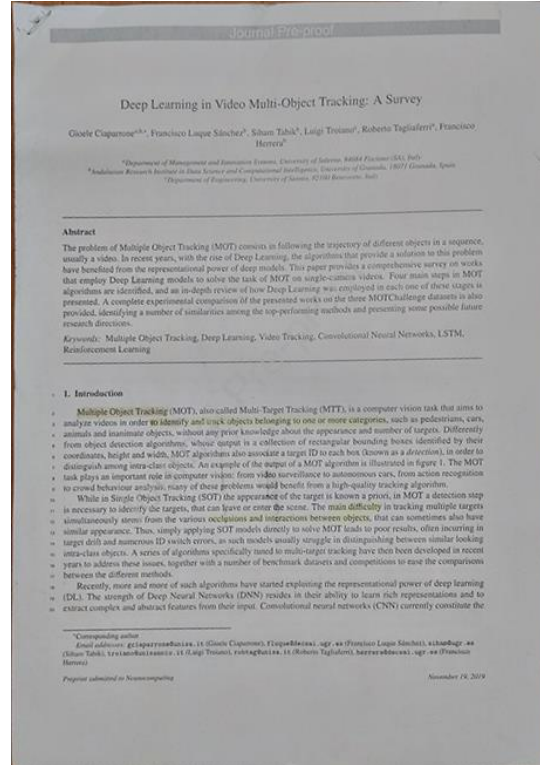
روش های بهتر برای این تصویر:

5. تبدیل فضا به LAB و اعمال clahe (مقایسه با تصویر crop شده):



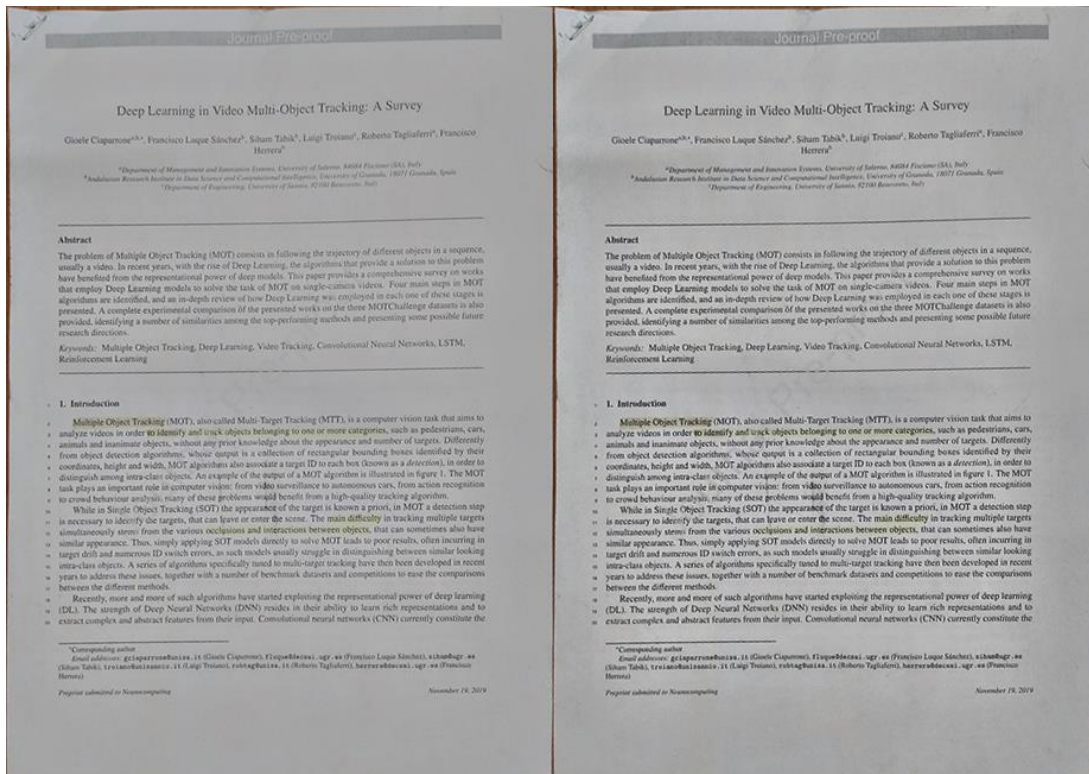
6. روش تقسیم بر فیلتر گاوسی: ابتدا، با اعمال یک فیلتر محو بسیار قوی بر روی تصویر اصلی، یک تصویر "fake" flat-field تولید می کنیم. سپس تصویر crop شده را در میانگین  $F$  ضرب می کنیم و تصویر حاصل را بر  $F$  (پیکسل به پیکسل) تقسیم می کنیم تا تصویر اصلاح شده  $C$  بدست آید. ضرب فقط برای حفظ روشنایی کلی است، اما بیشتر تغییرات در تقسیم اتفاق می افتد. در تصویر اصلاح شده بسیاری از نور و رنگ های ناخواسته در مقیاس بزرگ حذف شده اند (مثل سایه ها).





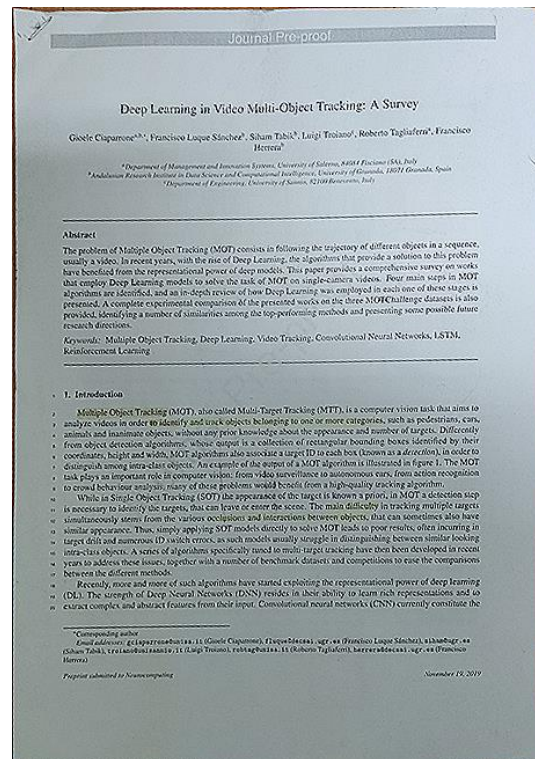
می بینیم که سایه بزرگ پایین تصویر حذف شده است.

7. ترکیب روش بالا و CLAHE روی فضای رنگی LAB:

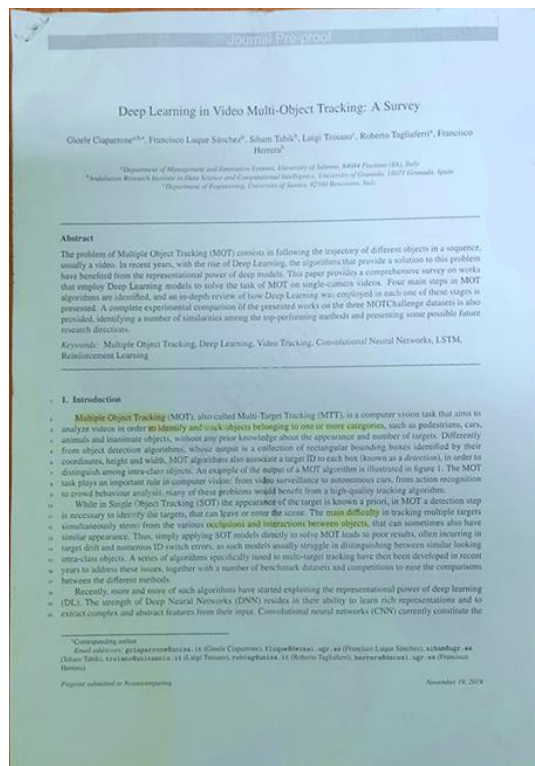




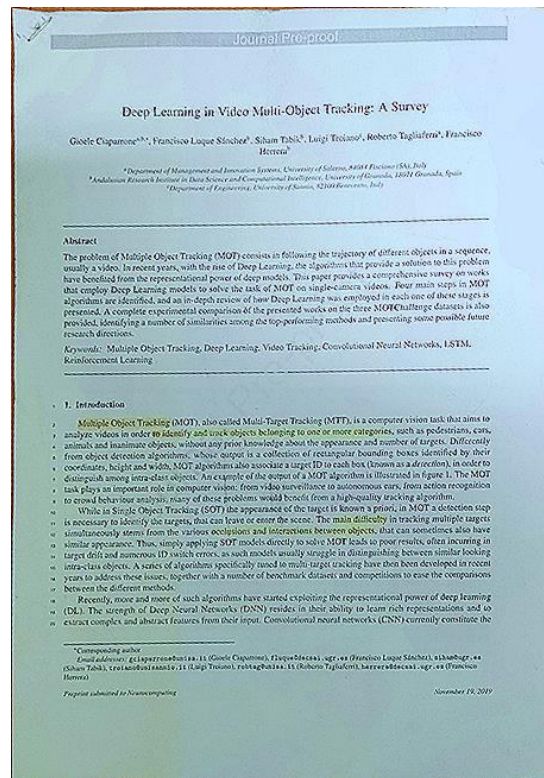
## 8. Sharp کردن تصویر:



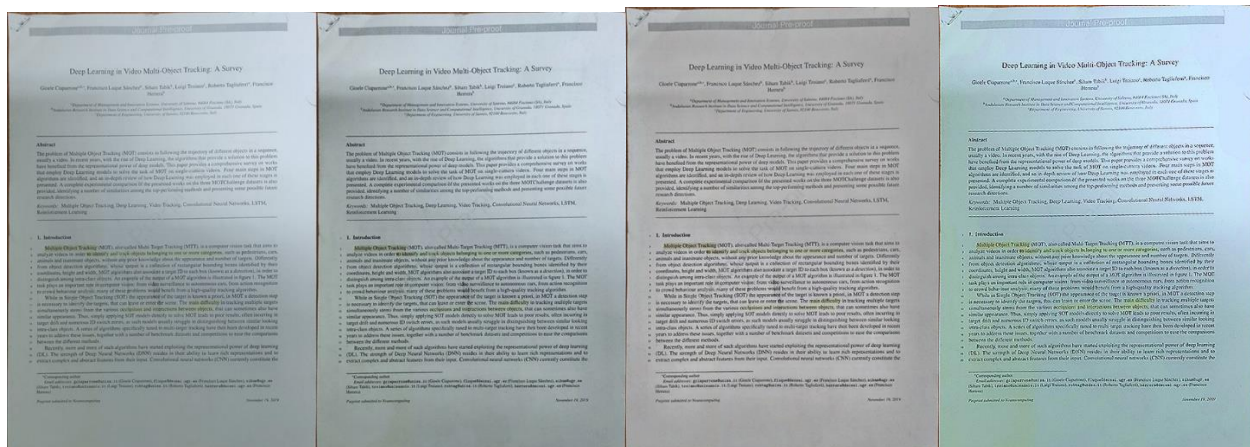
## 9. بالا بردن value و saturation تصویر:



## 10. ترکیب روش Sharp کردن تصویر و بالا بردن saturation و value:

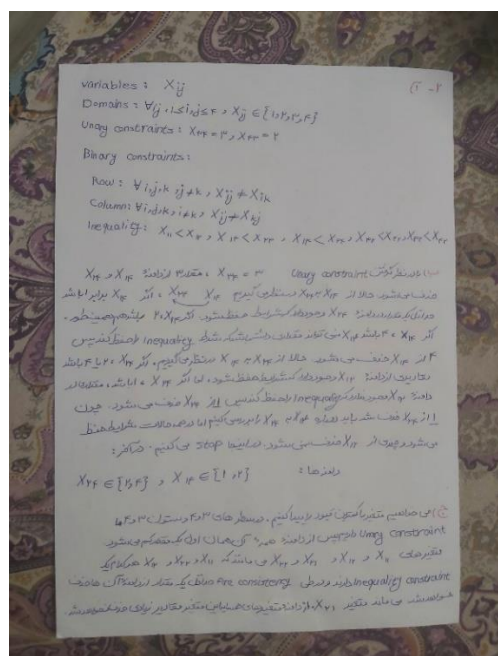


در کل، این روش ها را می توان به عنوان فیلتر های مختلف camscanner خود در نظر بگیریم که ممکن است هرکدام برای تصویر های مختلف خروجی های بهتر یا بدتری داشته باشند. بهترین خروجی ها برای این تصویر مربوط به روش های 4 و 6 و 8 است. مقایسه نتیجه این روش ها را در کنار تصویر crop شده اصلی می بینیم:

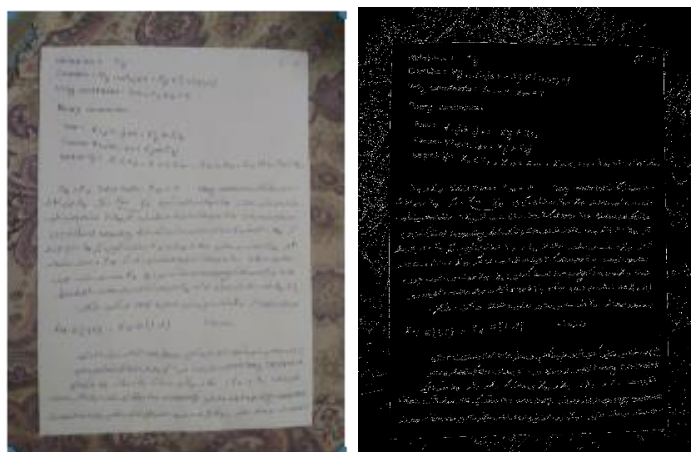


کد همه روش ها در نوت بوک موجود هست. اما در تابع enhance کد روش آخر را قرار دادیم که نتیجه بهتری برای این تصویر داشت.

7- **پارت امتیازی:** برای امتحان کردن نحوه کارکرد برنامه تصویری که خودم گرفتم را به برنامه می‌دهم. این تصویر در پوشه images با نام example2.jpg ذخیره شده است.



نتیجه مراحل تشخیص لیه و تشخیص رئوس برای این تصویر:

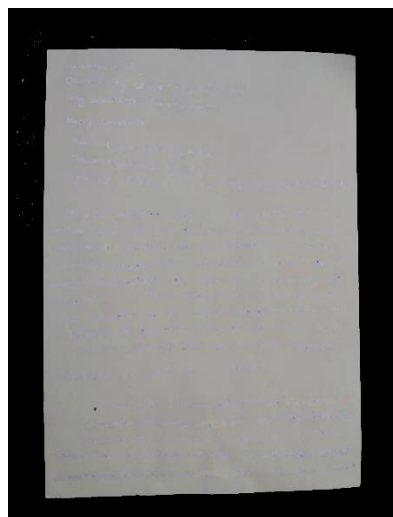


می‌بینیم که نقاط گوشه برگه به دلیل رنگ و نویز زیاد بک گراند پیدا نشدند. و در واقع همان نقاط اولیه (نقاط گوشه خود تصویر) که در ابتدا به آن مقداردهی کرده بودم را دارد.

ابتدا سعی کردم با قویتر و ضعیفتر کردن فیلتر گاوسی یا bilateral و عوض کردن threshold های canny خروجی را بهبود دهم اما باز هم چارچوب صفحه درست لبه یابی نشد و نقاط گوشه برگه را تشخیص نداد.

## تغییر پیشنهادی در الگوریتم:

می‌توانیم از پردازش‌های مورفولوژی که به تازگی در درس یاد گرفتیم استفاده کنیم. (مربوط به بحث image segmentation). با استفاده از تابع `cv2.morphologyEx`، با انجام عملیاتی مانند `erosion` و `dilation`، می‌توانیم تبدیل‌های مورفولوژیکی پیشرفته‌ای را انجام دهیم. در اینجا، ما عملیات `closing` را انجام می‌دهیم. عملگر بسته (`closing`) برای حذف حفره‌های کوچک و هموار کردن محیط نواحی تعریف شده است. این عملگر ناحیه‌های سیاه که در احاطه پیکسل‌های سفید هستند را حذف می‌کند. با این کار محتویات برگه را حذف کرده و در تصویر به یک صفحه برگه سفید خالی می‌رسیم. گام بعدی حذف پس‌زمینه تصویر است. از `GrabCut` در `OpenCV` استفاده می‌کنیم. برای این کار نیاز به یک کادر محدود در اطراف جسم داریم، سپس هر چیزی که خارج از این `box` است، به عنوان پس‌زمینه در نظر گرفته می‌شود. `GrabCut` به طور خودکار ما را از شر تمام پس‌زمینه‌ها خلاص می‌کند، و تنها برگه ما باقی می‌ماند. پس 20 پیکسل گوشه را به عنوان پس‌زمینه می‌گیریم و `GrabCut` به طور خودکار پیش‌زمینه و پس‌زمینه را تعیین می‌کند و تنها برگه را برای ما باقی می‌گذارد. نتیجه:



پس از این مرحله همه مراحل قبلی را به ترتیب مانند قبل انجام می‌دهیم. در واقع این مرحله به عنوان مرحله اول اضافه شده است. حالا کار لبه یاب بسیار راحت تر است. نتیجه نقطه یابی:





(۱-۲)

variables :  $X_{ij}$ Domains :  $\forall ij, 1 \leq i, j \leq 4, X_{ij} \in \{1, 2, 3, 4\}$ Unary constraints :  $X_{34} = 3, X_{43} = 2$ 

Binary constraints :

Row :  $\forall i, j, k, j \neq k, X_{ij} \neq X_{ik}$ Column :  $\forall i, j, k, i \neq k, X_{ij} \neq X_{kj}$ Inequality :  $X_{11} < X_{12}, X_{13} < X_{23}, X_{14} < X_{24}, X_{32} < X_{22}, X_{32} < X_{42}, X_{34} < X_{44}$ 

ب) با در نظر گرفتن unary constraint  $X_{34} = 3$ ، مقدار از دامنه  $X_{14}$  و  $X_{24}$  حذف می‌شود. حالا از  $X_{14}$  و  $X_{24}$  در منطقی بگیریم  $X_{14}$  و  $X_{24}$ ، اگر  $X_{14}$  برابر باشد حداقل یک مقدار در دامنه  $X_{24}$  وجود دارد که شرایط حفظ شود. اگر  $X_{14} = 2$  باشد هم به خط ۵. اگر  $X_{14} = 4$  باشد  $X_{12}$  نمی‌تواند مقداری داشته باشد که شرط Inequality را حفظ کند پس ۴ از  $X_{12}$  حذف می‌شود. حالا از  $X_{14}$  به  $X_{12}$  در منطقی بگیریم، اگر  $X_{14} = 2$  یا  $4$  باشد مقایسه از دامنه  $X_{12}$  وجود دارد که شرایط حفظ شود، اما اگر  $X_{14} = 3$  باشد، مقایسه از دامنه  $X_{12}$  وجود ندارد که Inequality را حفظ کند پس از  $X_{14}$  حذف می‌شود. چون ۱ از  $X_{14}$  حذف شد باید دوباره  $X_{14}$  به  $X_{12}$  را بررسی کنیم اما در همه حالات شرایط حفظ می‌شود و چیزی از  $X_{12}$  حذف نمی‌شود. در اینجا stop می‌کنیم. در آخر:

دامنه‌ها :  $X_{14} \in \{1, 2\}$  و  $X_{24} \in \{2, 4\}$

ج) می‌خواهیم متغیر با کمترین قیود را پیدا کنیم. در سطرهای ۳ و ۴ و ستون ۳ و ۴ ما unary constraint داریم پس از دامنه همه آن همان اول یک مقدار کم می‌شود. متغیرهای  $X_{11}$  و  $X_{12}$  و  $X_{13}$  و  $X_{23}$  می‌مانند که  $X_{11} < X_{23}$  و  $X_{12} < X_{22}$  و هر کدام یک Inequality constraint دارند و در طی Are consistency حداقل یک مقدار از دامنه آن‌ها حذف خواهد شد. می‌ماند متغیر  $X_{21}$  از دامنه متغیرهای همسایه این متغیر مقایسه زیادی حذف خواهد شد.

variables :  $X_{ij}$

(۱ - ۲)

Domains :  $\forall ij, 1 \leq i, j \leq 4, X_{ij} \in \{1, 2, 3, 4\}$

Unary constraints :  $X_{۳۴} = ۳, X_{۴۳} = ۲$

Binary constraints :

Row :  $\forall i, j, k, j \neq k, X_{ij} \neq X_{ik}$

Column :  $\forall i, j, k, i \neq k, X_{ij} \neq X_{kj}$

Inequality :  $X_{۱۱} < X_{۱۲}, X_{۱۳} < X_{۲۳}, X_{۱۴} < X_{۲۴}, X_{۳۲} < X_{۲۲}, X_{۳۲} < X_{۴۲}$

ب) با در نظر گرفتن unary constraint  $X_{۳۴} = ۳$ ، مقدار ۳ از دامنه  $X_{۳۴}$  و  $X_{۱۴}$  حذف می شود. حالا از  $X_{۳۴}$  به  $X_{۱۴}$  در نظری می گیریم  $X_{۳۴}$ ، اگر  $X_{۳۴}$  برابر باشد حداقل یک مقدار در دامنه  $X_{۲۴}$  وجود دارد که شرایط حفظ شود. اگر  $X_{۱۴}$  ۲ باشد هم همین طور. اگر  $X_{۱۴}$  ۴ باشد  $X_{۱۴}$  نمی تواند مقداری داشته باشد که شرط Inequality را حفظ کند پس ۴ از  $X_{۱۴}$  حذف می شود. حالا از  $X_{۱۴}$  به  $X_{۱۲}$  در نظری می گیریم. اگر  $X_{۱۲}$  ۲ یا ۴ باشد مقایسه از دامنه  $X_{۱۲}$  وجود دارد که شرایط حفظ شود، اما اگر  $X_{۱۲}$  ۱ باشد، مقایسه از دامنه  $X_{۱۲}$  وجود ندارد که Inequality را حفظ کند پس ۱ از  $X_{۱۲}$  حذف می شود. چون ۱ از  $X_{۱۲}$  حذف شد باید دوباره  $X_{۱۴}$  به  $X_{۱۲}$  را بررسی کنیم اما در همه حالات شرایط حفظ می شود و چیزی از  $X_{۱۲}$  حذف نمی شود. در اینجا stop می کنیم. در آخر:

دامنه ها :  $X_{۲۴} \in \{۲, ۴\}$  و  $X_{۱۴} \in \{۱, ۲\}$

ج) می خواهیم متغیر با کمترین مقدار را پیدا کنیم. در سطر های ۳ و ۴ و ستون ۳ و ۴ ما unary constraint داریم پس از دامنه همه آن همان اول یک مقدار کم می شود. متغیر های  $X_{۱۱}$  و  $X_{۱۲}$  و  $X_{۲۱}$  و  $X_{۲۲}$  می مانند که  $X_{۱۱}$  و  $X_{۲۲}$  و  $X_{۲۱}$  هر کدام یک Inequality constraint دارند و در طی Are consistency حداقل یک مقدار از دامنه آن ها حذف خواهد شد می مانند متغیر  $X_{۲۱}$  از دامنه متغیر های همای این متغیر مقادیر زیادی حذف خواهد شد.



