داکیومنت تکلیف شماره ۲

■ سوال ۱

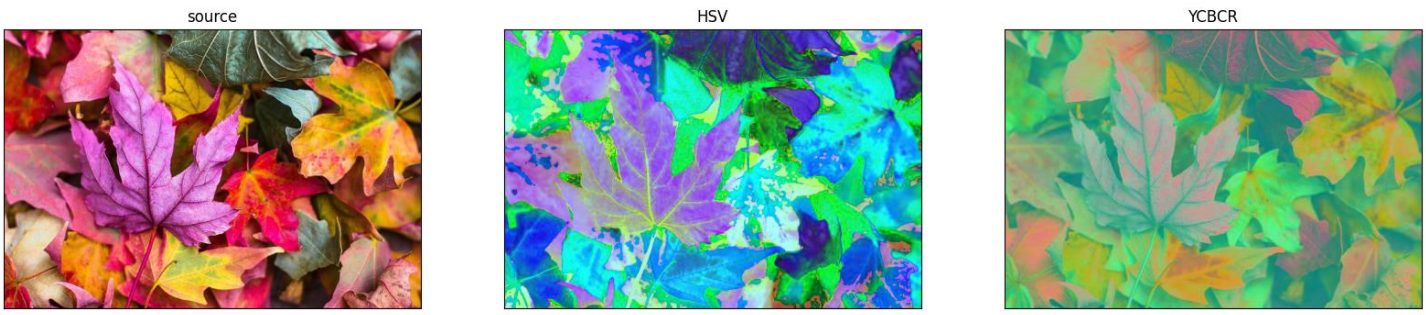
الف)

```python
import numpy as np

# Define the RGB color
rgb_color = np.array([50, 70, 130], dtype=np.uint8)

def RGB2CMYK(rgb_color):
    R, G, B = rgb_color / 255
    K = 1 - max(R, G, B)
    C = (1 - R - K) / (1 - K)
    M = (1 - G - K) / (1 - K)
    Y = (1 - B - K) / (1 - K)
    return np.array([C, M, Y, K])

CMYK_color = RGB2CMYK(rgb_color)
print(CMYK_color)
```

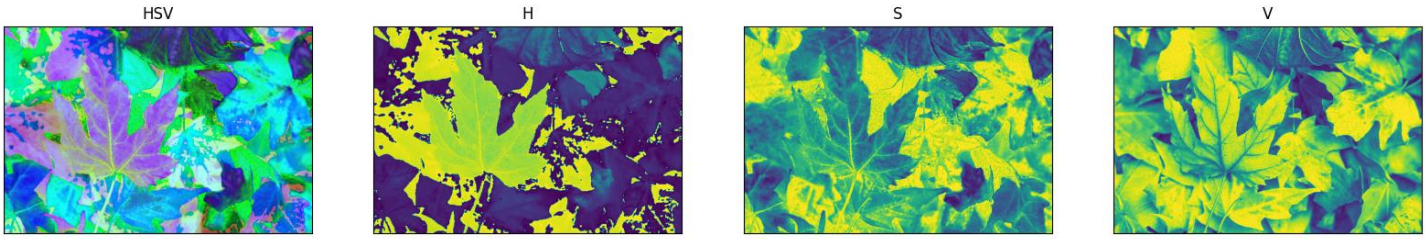
این کد یک تابع با نام RGB2CMYK است که مقداری رنگ RGB به عنوان ورودی دریافت می کند. در این تابع، رنگ RGB در ابتدا به مقادیر معادل در بازه ۰ تا ۱ تقسیم می شود. سپس مقدار کلیدی K که برای CMYK استفاده می شود، به عنوان ماکسیمم مقدار از مقادیر R ، Gو B محاسبه می شود. سپس مقادیر C ، Mو Y با توجه به مقادیر R، G، Bو K محاسبه می شوند. در نهایت، یک آرایه چهارتایی شامل مقادیر CMYK به عنوان خروجی تابع بازگردانده می شود.

در این کد، یک رنگ RGB با مقادیر (۵۰، ۷۰، ۱۳۰) تعریف شده و سپس تابع RGB2CMYK با این مقدار به عنوان ورودی فراخوانی شده است. نتیجه به دست آمده برای CMYK رنگ با استفاده از مقدار ورودیRGB ، در خط آخر با استفاده از تابع پرینت چاپ می شود.

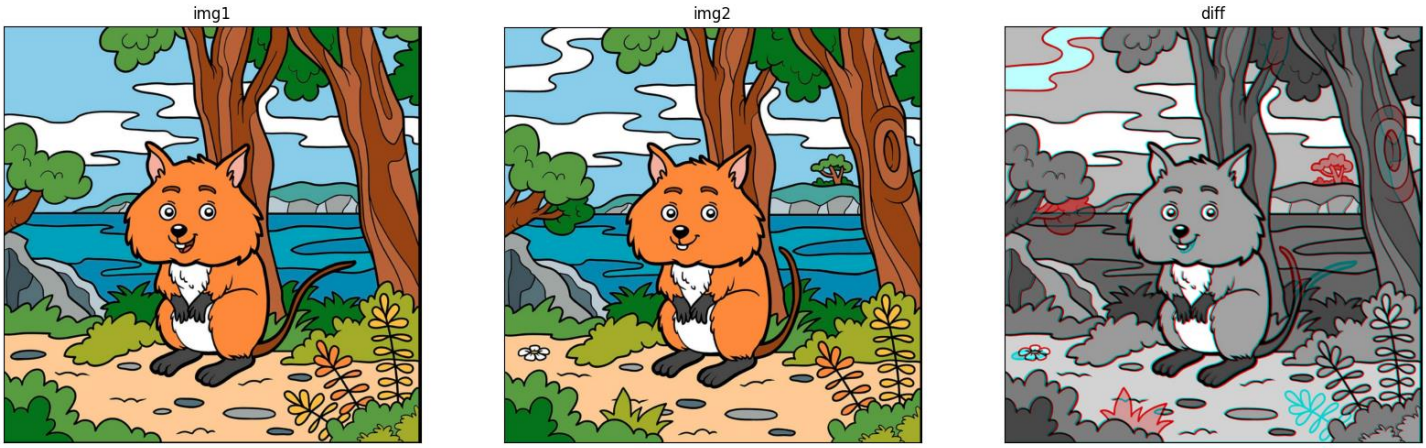
ب)



ج)



د) این کد یک تابع به نام **dif**دارد که دو تصویر ورودی را گرفته، آن‌ها را با یکدیگر مقایسه کرده و تصویری را بازگردانده که تفاوت بین آن دو را نشان می‌دهد. برای این کار، این تابع ابتدا تصاویر ورودی را از فضای رنگی **BGR** به **RGB** تغییر می‌دهد و سپس تصویر دوم را با استفاده از تابع **resize**تغییر اندازه می‌دهد تا اندازه آن با تصویر اول یکسان شود. سپس تصویری جدید با ابعاد مشابه تصویر اول ایجاد شده و همه پیکسل‌های آن به مقدار صفر تنظیم می‌شوند. در نهایت، کانال‌های قرمز و سبز تصویر اول و دوم به ترتیب در تصویر جدید کپی شده و کانال آبی تصویر دوم نیز در تصویر جدید کپی می‌شود. سپس تصویر نهایی با تفاوت بین دو تصویر، تابع خروجی داده می‌شود.



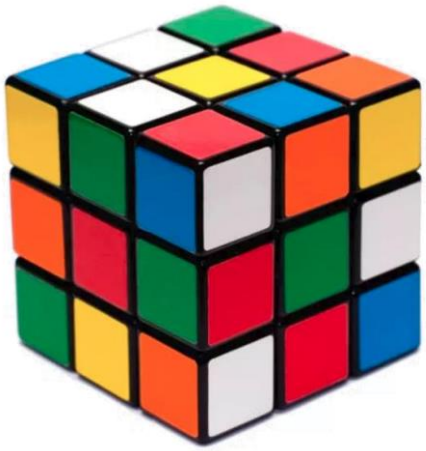
ه) علاوه بر اینکه فضای رنگی **gray** اطلاعات کمتری در مورد هر پیکسل در اختیار ما قرار می‌دهد، چندین دلیل دیگر نیز وجود دارد که استفاده از فضای رنگی مناسب در پردازش تصویر بسیار مهم است:

۱. تفاوت رنگ‌ها: در فضای رنگی**gray** ، رنگ‌ها تنها با سطح خاکستری پیکسل‌ها مشخص می‌شوند و اگر بخواهیم تفاوت رنگ‌ها را تفسیر کنیم، نیازمند محاسبات پیچیده‌تری هستیم. از طرفی در فضای رنگی **RGB**، هر رنگ با سه مقدار قرمز، سبز و آبی به صورت مجزا مشخص می‌شود و تفاوت رنگ‌ها بسیار واضح است.

۲. تحلیل روشنایی: در برخی از کاربردهای پردازش تصویر مثل تشخیص اجسام در تصاویر، تحلیل روشنایی بسیار مهم است. در فضای رنگی**gray** ، این تحلیل بسیار محدود است و ممکن است اطلاعات مهمی را از دست بدهیم. در فضای رنگی**HSV** ، می‌توانیم روشنایی را با سه مقدار سطوح روشنایی، اشباع و رنگ تفکیک کنیم و برای برخی از کاربردها مانند تشخیص رنگ‌های خاص، این فضای رنگی بسیار مناسب است.

۳. تفسیر رنگ‌ها: در برخی از کاربردهای پردازش تصویر، تفسیر رنگ‌ها بسیار مهم است. در فضای رنگی**gray** ، تفسیر رنگ‌ها ناممکن است ولی در فضای رنگی**CMYK** ، می‌توانیم رنگ‌های چاپی را تفسیر کنیم و برای برخی از کاربردها مانند چاپ و تولید محصولات چاپی، این فضای رنگی بسیار مهم است.

برای مثال به دو تصویر زیر نگاه کنید. در حالت **gray**، رنگ های آبی و سبز و قرمز تقریبا یک رنگ می شوند و تشخیص آن ها از هم دشوار است و ممکن است در برخی از تصاویر این رنگ ها با یکدیگر ادغام شوند و مرز آن ها قابل تشخیص نباشند.



■ سوال ۲

این کد برای ادغام چند تصویر با یکدیگر به کار می‌رود. ابتدا یک شی از کلاس **Stitcher** در **OpenCV** ایجاد می‌شود و سپس این شی با استفاده از تصاویر ورودی، تصویر جدیدی را با استفاده از روش ادغام تصاویر ایجاد می‌کند. اگر مقدار نسخه **OpenCV** برابر با ۳ باشد، تابع **createStitcher()** استفاده می‌شود و در غیر این صورت تابع **Stitcher_create()** به کار می‌رود. پس از اجرای این کد، تصویر جدید ادغام شده در متغیر **stitched_image**ذخیره می‌شود و وضعیت عملیات ادغام تصاویر نیز در متغیر **status** بازگردانده می‌شود.

■ **سوال ۳**

الف) این کد یک تابع با نام put_mask است که عکس چهره و عکس ماسک را دریافت می‌کند و ماسک را روی چهره اعمال می‌کند و عکس حاصل را برمی‌گرداند.

اولین قدم این است که با استفاده از کتابخانه dlib و فایل shape_predictor_68_face_landmarks.dat چهره در تصویر تشخیص داده می‌شود. سپس با استفاده از نقاط landmark چهره، نقاط متناظر با بینی، پوزخند چپ، پوزخند راست و دُهن را انتخاب می‌کنیم.

سپس بردارهایی به نام face_points و mask_points تعریف می‌شوند که نقاط متناظر با چهره در تصویر و نقاط متناظر با ماسک را دربرمی‌گیرند.

سپس با استفاده از تابع getPerspectiveTransform ، ماتریس تبدیلی برای تبدیل نقاط متناظر با ماسک به نقاط متناظر با چهره به دست می‌آید.

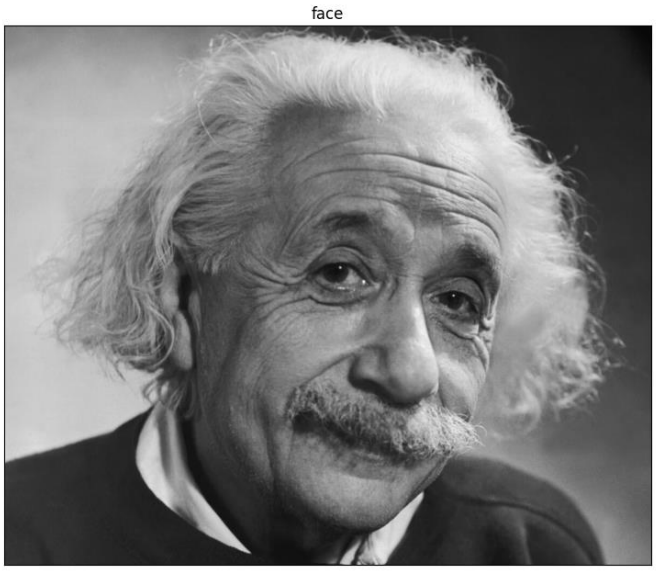
سپس با استفاده از تابع warpPerspective ، ماسک به صورت projective بر روی چهره اعمال می‌شود و تصویر جدیدی به نام mask_warped به دست می‌آید.

سپس با استفاده از تابع cvtColor ، تصویر mask_warped به یک تصویر خاکستری تبدیل می‌شود. سپس با استفاده از تابع threshold ، تصویر خاکستری به یک تصویر دودویی تبدیل می‌شود و تصویر نهایی ماسک به دست می‌آید.

در مرحله بعد، تصویر نهایی چهره به دست می‌آید. در این بخش از کد، ابتدا با استفاده از تابع cv2.bitwise_not، ماسک را به‌صورت عکسِ تهیه می‌کنیم. سپس با استفاده از تابع cv2.bitwise_and، عکس چهره (face_img) را با نقاطی که در ماسک خالی نیستند (mask_rgb) یا به عبارتی بخشی از ماسک که سفید است، عمل AND می‌کنیم. این کار باعث می‌شود که بخش‌های چهره که مشابه بخش‌های ماسک خالی نیستند، قابل دید نباشند.

سپس با تابع cv2.bitwise_and، ماسک ماسک وارپ شده (mask_warped) را با ماسک خالی نیست (mask_rgb)عمل ANDمی‌کنیم. این کار باعث می‌شود که بخش‌های ماسک که سفید هستند، بر روی بخش‌های مشابه در عکس چهره نمایش داده شوند.

در نهایت با استفاده از تابع cv2.add، عکس چهره که در آن بخش‌های ماسک با بخش‌های مشابه در عکس چهره جایگزین شده‌اند (result) را با ماسکِ وارپ شده (mask_warped_masked) جمع می‌کنیم و نتیجه را برمی‌گردانیم.



ب) تشخیص landmark یا نقاط قابل توجه روی چهره، یکی از مهمترین و پرکاربردترین مسائل در بینایی کامپیوتر و تشخیص چهره است Landmark .ها به عنوان نقاط مشخصی که بر روی چهره وجود دارند، مانند چشم، بینی، لب، گوش و... تعریف می شوند. تشخیص این landmark ها، به صورت خودکار، به دلیل دقت بالا و کاربردهای گسترده، از اهمیت بسیاری برخوردار است. برای تشخیص landmark های چهره، می توان از چند روش استفاده کرد که در ادامه به توضیح یکی از این روش ها می پردازم.

یکی از روش های تشخیص landmark چهره، استفاده از شبکه های عصبی کانولوشنی (CNN) است. این روش در چند مرحله انجام می شود:

1. **تشخیص چهره :ابتدا باید چهره در تصویر شناسایی شود. برای این کار می توان از الگوریتم های تشخیص چهره مانند Haar Cascade و یا شبکه های عصبی مانند SSD استفاده کرد.**
2. **پیش پردازش :در این مرحله، تصویر چهره به صورت استاندارد سازی شده و اندازه آن کوچک شده تا سرعت پردازش افزایش یابد.**
3. **استخراج ویژگی :در این مرحله، شبکه عصبی کانولوشنی بر روی تصویر استفاده می شود تا ویژگی های مختلفی از تصویر استخراج شوند. به عبارت دیگر، شبکه عصبی با عبور تصویر از آن، به صورت خودکار، اجزای مختلف چهره را شناسایی می کند.**
4. **در این مرحله، با استفاده از اطلاعات به دست آمده در مرحله قبلی، به طور دقیق landmark های مختلف روی چهره تشخیص داده می شوند. این کار با استفاده از یک شبکه عصبی کانولوشنی و یا یک شبکه عصبی مولد (Generative Adversarial Network) انجام می شود. در نهایت، مختصات landmark های تشخیص داده شده بر روی تصویر چهره قرار داده می شوند تا بتوان از آنها برای کاربردهای مختلفی مانند تشخیص احساسات صورت، تشخیص جنسیت، تشخیص سن و... استفاده کرد.**

■ **سوال ۴**

الف)

این کد با استفاده از کتابخانه OpenCV ، تصویر ورودی را ابتدا به grayscale تبدیل می‌کند تا با کاهش ابعاد تصویر، سرعت پردازش آن افزایش پیدا کند.

سپس با استفاده از یک فیلتر میانه (median filter) نویز تصویر کاهش داده می‌شود تا در مراحل بعدی پردازش دقیق‌تر صورت گیرد. در این کد، از یک فیلتر ۵x۵ میانه استفاده شده است.

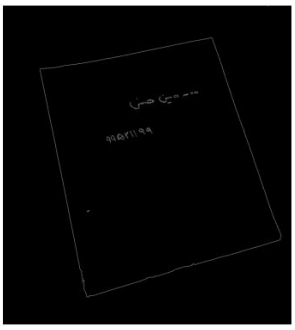
با اعمال فیلتر گائوسی (Gaussian blur) بر روی تصویر، نویزهای بیشتری از تصویر حذف می‌شود. این فیلتر با استفاده از یک ماتریس یکنواخت یا همان kernel با اندازه دلخواه بر روی تصویر اعمال می‌شود. در این کد، از یک kernel ۱x۱ استفاده شده است.

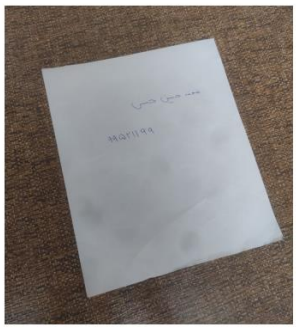
سپس با استفاده از تابع thresholding ، مقادیر روشنی پیکسل‌ها بر اساس یک آستانه تعیین شده، به مقادیر دلخواه (۰ یا ۲۵۵) تبدیل می‌شوند. در این کد، آستانه برابر با ۱۳۰ قرار داده شده است.

در ادامه با استفاده از تابع erode و dilate ، نویزهای موجود در تصویر حذف می‌شوند. این دو تابع به ترتیب، عملیات خرد کردن و توریفی کردن را بر روی تصویر اعمال می‌کنند. در این کد، یک kernel ۵x۵ به عنوان نواحی خرد شده و توریفی شده در نظر گرفته شده است.

یک بار دیگر نیز این عملگر ها روی تصویر با کرنلی بزرگتر (۱۵*۱۵) اهمال می شوند تا فقط لبه های برگه آچار مشخص شود.

در نهایت، با استفاده از روش Canny Edge Detection ، لبه‌های تصویر شناسایی می‌شوند. در این روش، ابتدا با استفاده از روش Sobel ، گرادیان تصویر محاسبه می‌شود و سپس با اعمال یک آستانه تعیین شده بر روی گرادیان، لبه‌های تصویر با استفاده از مشتقات گرادیان تشخیص داده می‌شوند. در این کد، آستانه‌های برابر با ۵۰ و ۱۵۰ برای تعیین لبه‌ها استفاده شده است. این پارامترها به ترتیب حداقل و حداکثر مقدار گرادیان برای شناسایی لبه‌ها را مشخص می‌کنند. در نهایت، نتیجه‌ی تصویر حاصل شده با استفاده از توابع imshow و subplot نمایش داده می‌شود.



ب) این کد، با استفاده از تابع cv2.findContours، لبه‌های تصویر ثابت شده در تصویر دودویی thresh2را پیدا می‌کند و با استفاده از تابع cv2.drawContoursآن‌ها را بر روی یک کپی از تصویر اصلی به رنگ سبز رسم می‌کند.

در تابع cv2.findContours، سه پارامتر ورودی وجود دارد. اولین پارامتر، تصویر دودویی است که باید لبه‌های آن پیدا شود. دومین پارامتر نحوه‌ی جستجو در لبه‌های یافت شده را مشخص می‌کند. پارامتر cv2.RETR_TREEبه معنای بازگشت درختی است و به معنای برگشت تمامی لبه‌ها به علاوه تعلق و وابستگی آن‌ها به یکدیگر است. سومین پارامتر نوع روش تقریب محاسبه‌ی لبه‌ها را مشخص می‌کند. پارامتر cv2.CHAIN_APPROX_SIMPLEبه معنای حذف انتهایی و تقریب ساده‌ی لبه‌ها است.

در تابع cv2.drawContours، سه پارامتر ورودی وجود دارد. اولین پارامتر، تصویری است که روی آن لبه‌ها به شکل کنتور رسم می‌شود. دومین پارامتر، کنتورهای یافت شده است که برای همه لبه‌های یافت شده 1-است. سومین پارامتر، رنگ کنتور رسم شده در تصویر است که در اینجا سبز است و ضخامت خط آن ۱۰ پیکسل است.

در نهایت، تصویر حاصل با استفاده از تابع imshowنمایش داده می‌شود.

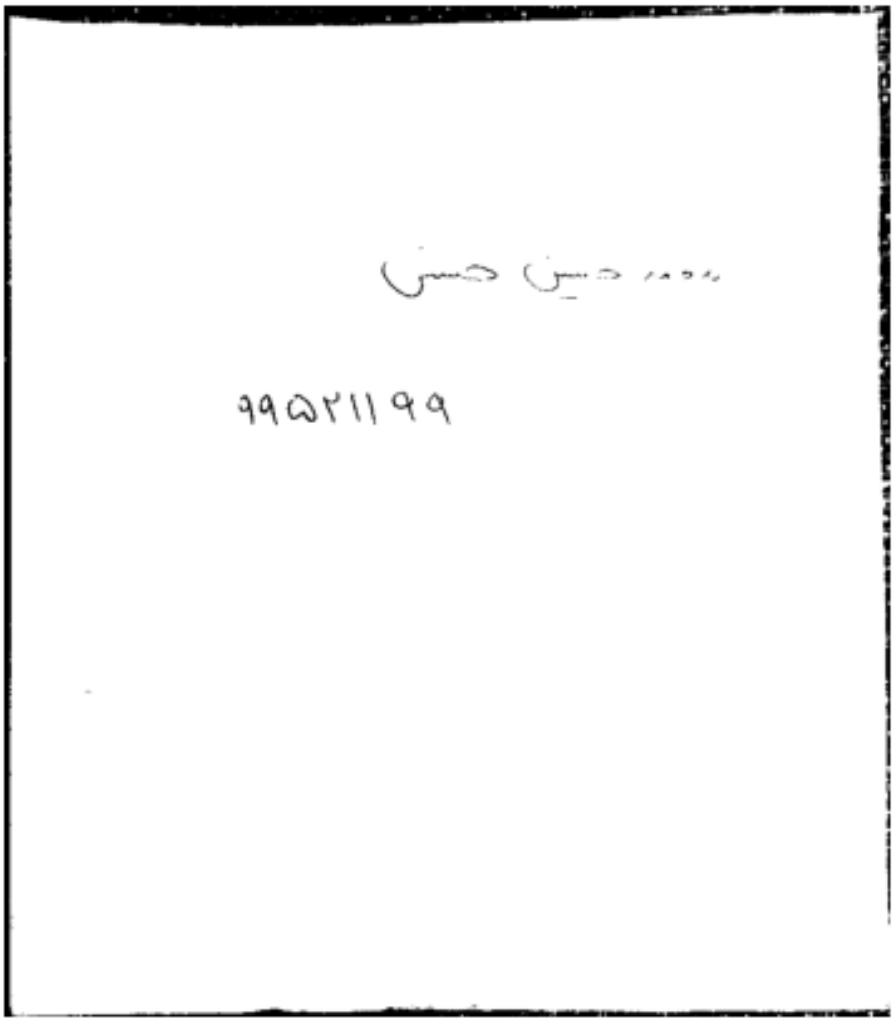
ج) در این بخش، تصویر ورودی که پس از یافتن چند طرح برش داده شده است، به یک تصویر خطی با ابعاد منظم تبدیل می‌شود.

ابتدا محدوده مستطیلی که کاغذ در آن قرار دارد، با استفاده از تقریب پلی‌گانون، به دست می‌آید. سپس نقاط گوشه‌های مستطیل با استفاده از تابع reorderدر ترتیب قرارگیری صحیح قرار می‌گیرند.

در مرحله بعد، با توجه به اندازه و شکل مستطیل، ماتریس تبدیل افقی و عمودی مورد نیاز برای چیدمان مجدد تصویر (perspective transform) محاسبه می‌شود. این ماتریس توسط تابع cv2.getPerspectiveTransform محاسبه می‌شود.

سپس تصویر اولیه با استفاده از ماتریس تبدیل به یک تصویر جدید با اندازه و شکل مناسب تبدیل می‌شود. در این مرحله، تابع cv2.warpPerspectiveبه کار گرفته می‌شود.

سپس با اعمال یک threshold ساده، تصویر خطی به تصویر خاکستری تبدیل شده و در نهایت در تصویر خاکستری خروجی نمایش داده می‌شود.



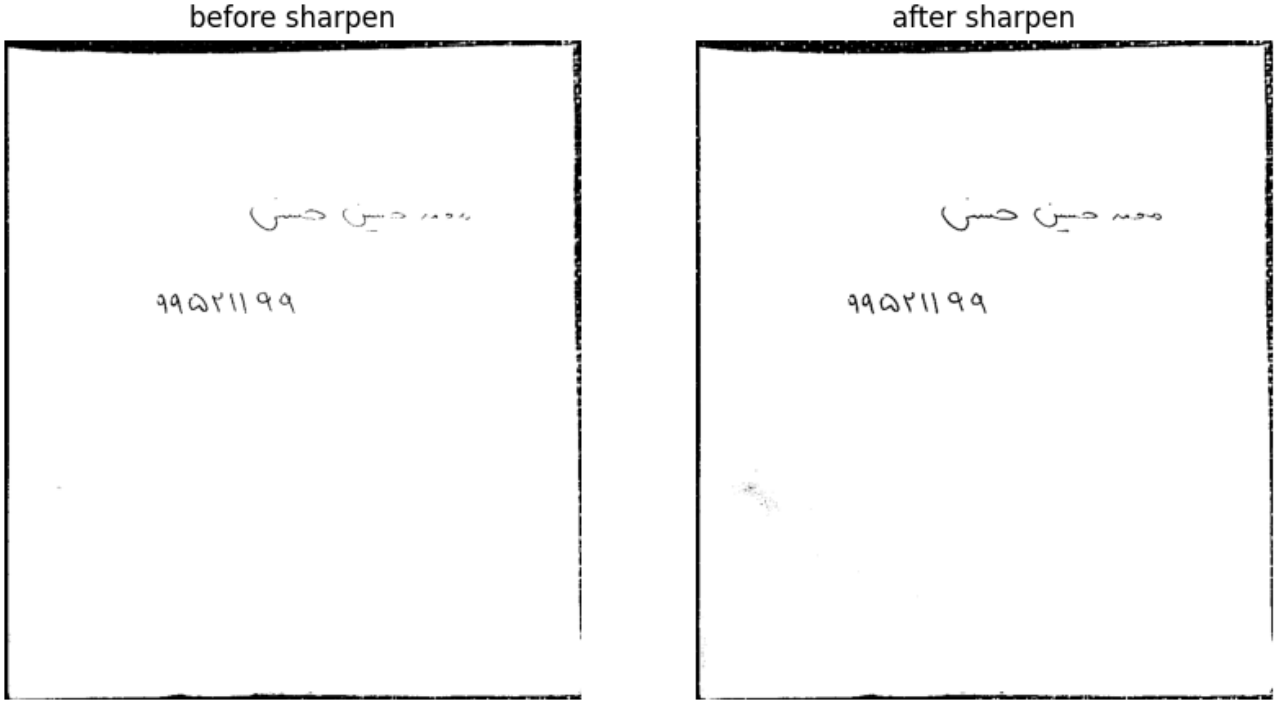
د) با توجه به کد این بخش، چندین روش برای بهبود کیفیت تصویر کاغذ وجود دارد که در زیر آن‌ها را مرور می‌کنیم:

1. **اعمال فیلتر گاوسی: این فیلتر با حذف نویزهایی که در تصویر وجود دارد، کیفیت تصویر را بهبود می‌بخشد.**
2. **تیز کردن تصویر: در این روش، با استفاده از یک کرنل خاص، تصویر تیزتر و واضح‌تر می‌شود.**
3. **تنظیم روشنایی و کنتراست: در این روش، با تنظیم روشنایی و کنتراست تصویر، آن را بهبود می‌بخشیم.**
4. **حذف نویز با استفاده از فیلتر مدین: در این روش، با حذف نویزهای موجود در تصویر، کیفیت تصویر بهبود می‌یابد.**

با توجه به این که تصویر کاغذ ما در بخش قبل با استفاده از تبدیل هاف و شناسایی چهارچوب آن، استخراج شده است، روش‌هایی که می‌توان برای بهبود کیفیت تصویر کاغذ استفاده کرد، اعمال فیلتر گاوسی و تیز کردن تصویر هستند.

در این روش، ابتدا با استفاده از فیلتر گائوسین، تصویر را نرم می‌کنیم تا نویزهای موجود در آن کاهش یابند. سپس با استفاده از یک کرنل شارپنینگ، تصویر را تیز تر می‌کنیم تا ویژگی‌های موجود در آن بهبود یابند. در نهایت، با اعمال یک تبدیل آستانه‌ای، تصویر به صورت دودویی تبدیل شده و کیفیت آن بهبود یافته است.

در کد ارائه شده، ابتدا با استفاده از فیلتر گائوسین، تصویر را نرم می‌کنیم. سپس با استفاده از یک کرنل شارپنینگ، تصویر را تیز تر می‌کنیم. در نهایت، با اعمال یک تبدیل آستانه‌ای، تصویر به صورت دودویی تبدیل شده و کیفیت آن بهبود یافته است. تصاویر قبل و بعد از اعمال این روش نیز در دو subplot مختلف نمایش داده شده‌اند.



■ سوال ۵

الف) الگوریتم هریس یک روش محاسباتی برای تشخیص نقاط ویژه در تصاویر دو بعدی است. در این الگوریتم، ابتدا گرادیان تصویر در جهت x و y محاسبه می‌شود. سپس حاصلضرب گرادیان در هر نقطه محاسبه می‌شود تا شدت تغییرات محلی در تصویر مشخص شود. برای افزایش دقت تشخیص نقاط ویژه، ماتریس حاصلضرب‌ها با یک فیلتر گاوسی کانولوشن می‌شود تا جمع حاصلضرب‌ها در یک پنجره به دست آید.

با استفاده از جمع حاصلضرب‌ها، تابع پاسخ هریس محاسبه می‌شود. این تابع، به وسیله تحلیل ساختار نقاط، نقاط گوشه‌ای را در تصویر تشخیص می‌دهد. برای این منظور، از یک آستانه استفاده می‌شود تا نقاطی که شدت تغییرات آنها از حد آستانه بیشتر است، به عنوان نقاط گوشه‌ای در نظر گرفته شوند.

بنابراین، الگوریتم هریس با استفاده از محاسبه مشخصات گرادیان و تحلیل ساختار نقاط، نقاط گوشه‌ای در تصویر را پیدا می‌کند. این روش به عنوان یکی از روش‌های مهم در بینایی ماشین استفاده می‌شود و در بسیاری از برنامه‌های کاربردی از جمله تشخیص چهره، ردیابی اشیا و پردازش تصویر استفاده می‌شود.

ب) در بخش پیاده سازی دستی هریس، تابع harris_points برای تشخیص گوشه های تصویر ورودی با استفاده از روش هاریس ایجاد شده است. این تابع با گرفتن تصویر ورودی به عنوان یک آرایه numpy ، تصویر گوشه را برمی گرداند.

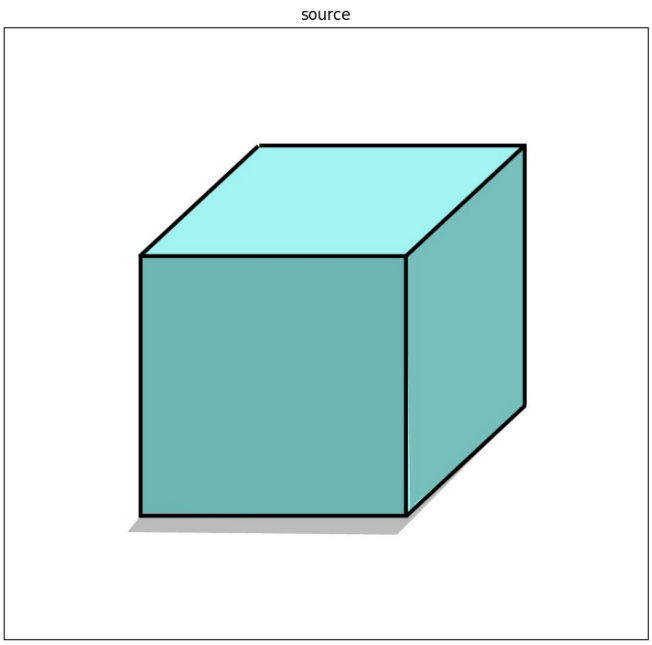
اولین کاری که تابع انجام می دهد ، تبدیل تصویر ورودی به تصویر سیاه و سفید است. سپس مشتقات تصویر با استفاده از اپراتور سوبل محاسبه می شوند. سپس محصولات مشتقات(Ixx ، Ixy و Iyy) محاسبه می شوند.

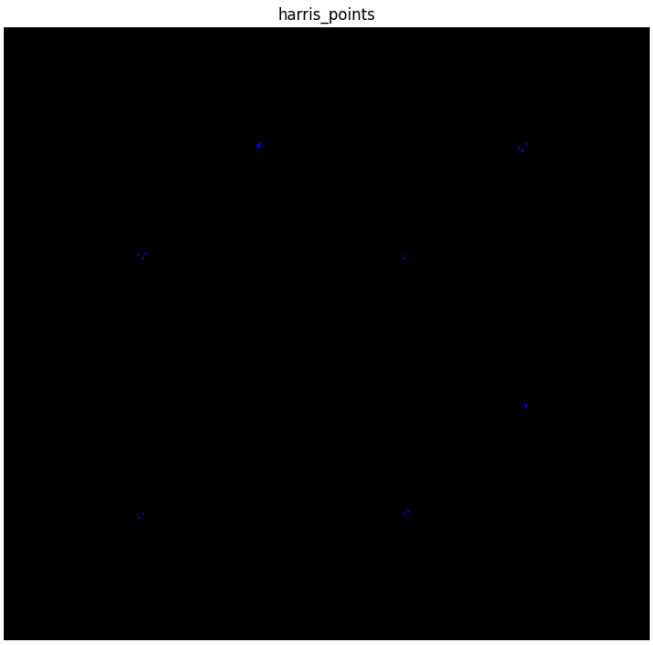
سپس یک فیلتر گاوسی بر روی محصولات مشتقات اعمال می شود تا نویز را کاهش دهد و سپس تابع پاسخ هاریس با استفاده از محصولات مشتقات محاسبه می شود. در نهایت، با اعمال آستانه بر روی تصویر پاسخ، تصویر گوشه ایجاد می شود.

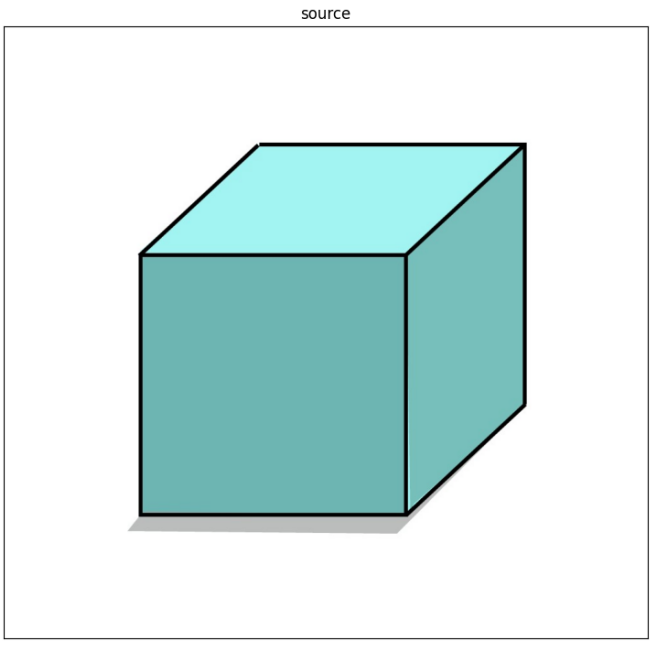
تابع harris_points_OpenCV نیز برای تشخیص گوشه های تصویر با استفاده از الگوریتم تشخیص گوشه هاریس ایجاد شده است، اما در این حالت از تابع cv2.cornerHarris در کتابخانه OpenCV برای محاسبه پاسخ هاریس استفاده می شود.

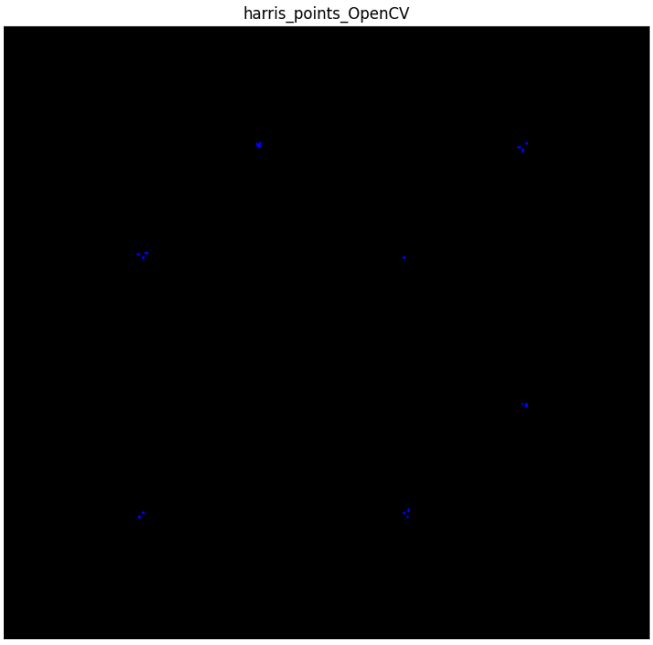
ابتدا تصویر ورودی به تصویر سیاه و سفید تبدیل شده و به نوع داده اعشاری (float32) تبدیل می شود. سپس با استفاده از تابع cv2.cornerHarris، پاسخ هاریس برای تصویر سیاه و سفید محاسبه می شود. سپس تصویر پاسخ با یک تابع گسترش داده می شود تا گوشه ها را به خوبی نشان دهد و با اعمال آستانه روی تصویر پاسخ، تصویر گوشه ایجاد می شود. در نهایت، تصویر گوشه به عنوان خروجی تابع برگردانده می شود.

مقایسه:









■ سوال ۶

روش‌های SIFT ، SURFو ORB همگی از روش‌هایی هستند که برای استخراج نقاط کلیدی در تصاویر استفاده می‌شوند. در ادامه، خلاصه‌ای از مقایسه این سه روش ارائه خواهد شد:

- SIFT

این روش در سال ۱۹۹۹ معرفی شد و برای استخراج نقاط کلیدی در تصاویر استفاده می‌شود. این روش برای تشخیص نقاط کلیدی از یک فیلتر گابور و دو روش ضرب ماتریس استفاده می‌کند. مزیت این روش، دقت بالا و قابلیت شناسایی نقاط کلیدی در شرایط نوری مختلف است.

- SURF

این روش در سال ۲۰۰۶ ارائه شد و برای استخراج نقاط کلیدی در تصاویر استفاده می‌شود. این روش از یک فیلتر گابور، ماتریس هسیان و روش‌های تشخیص لبه برای تشخیص نقاط کلیدی استفاده می‌کند. مزیت این روش، سرعت بالا و دقت بالایی که در شناسایی نقاط کلیدی در شرایط نوری مختلف دارد، است.

- ORB

این روش در سال ۲۰۱۱ ارائه شد و برای استخراج نقاط کلیدی در تصاویر استفاده می‌شود. این روش از یک الگوریتم مبتنی بر فیلتر گابور و تبدیل هاریس برای تشخیص نقاط کلیدی استفاده می‌کند. مزیت این روش، سرعت بالا و دقت بالایی که در شناسایی نقاط کلیدی دارد، است.

به طور کلی، روش SIFT دقت بالایی در شناسایی نقاط کلیدی در شرایط نوری مختلف دارد، روش SURF سریع‌تر و با دقت بالایی در شناسایی نقاط کلیدی در شرایط نوری مختلف است، و روش ORB سرعت بالا و دقت بالایی در شناسایی نقاط کلیدی دارد. با توجه به نیاز و شرایط مسئله، یکی از این روش‌ها ممکن است انتخاب مناسبی باشد.