به نام خدا

پروژه دوم

جبر خطی کاربردی - پاییز 1401

1.انجام دادن پروژه‌ها باید به صورت انفرادی صورت گیرد و در صورت مشاهده هرگونه تقلب نمره صفر برای کل پروژه منظور خواهد شد.

2. پاسخ ها مرتب و خوانا باشند.

3. در صورت وجود هرگونه ابهام، از طریق ایمیل تدریسیاری سوال خود را بپرسید: linearalgebra.fall1401@gmail.com

4. پاسخ خود را در یک فایل zip. با فرمت P?_Name_StudentNumber آپلود کنید.

موفق باشید. 😊

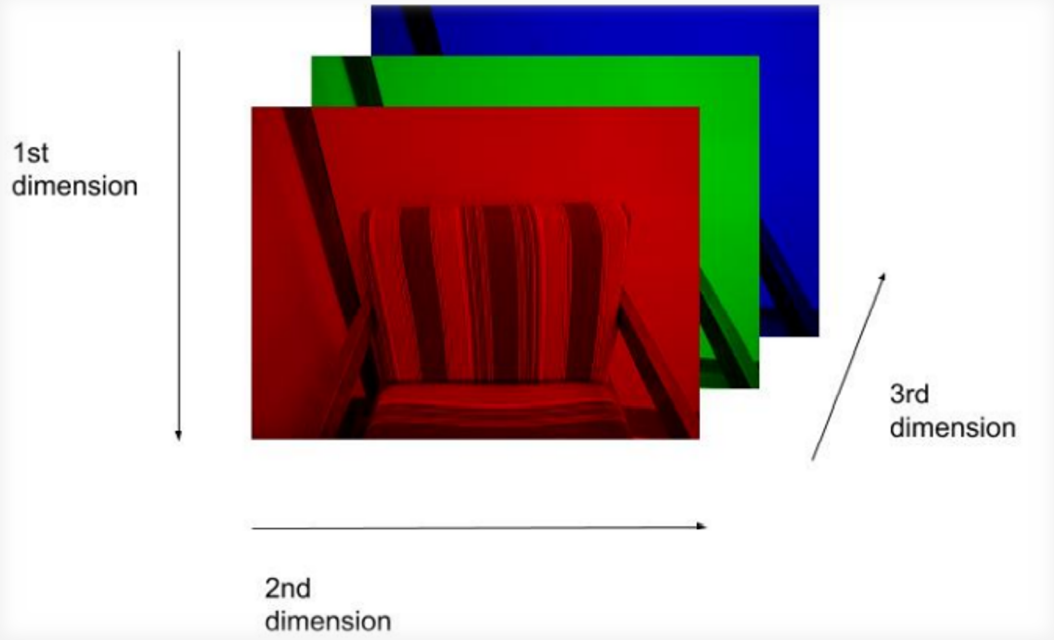
در این پروژه با توجه به مطالبی که درمورد تبدیل‌های خطی می‌دانیم، می‌خواهیم عملکرد برنامه‌های ادیت و فیلتر تصویر را شبیه‌سازی کنیم. ☺

اسنپ‌جبر

در این پروژه ابتدا با ساختار تصاویر RGB آشنا می‌شویم. سپس با استفاده از تبدیل‌های خطی، فیلترهایی از جمله سیاه-سفید کردن، برش، تغییر طول و عرض و شفاف‌سازی تصویر را پیاده‌سازی می‌کنیم. مراحل انجام پروژه در ادامه با جزئیات توضیح داده شده است.
امیدواریم با انجام این پروژه‌ی کاربردی، از یادگیری جبر خطی لذت ببرید. ☺

ساختار تصاویر RGB

در کامپیوتر ها، تصاویر رنگی در فرمت RGB هستند (به این معنی که این تصاویر دارای سه کانال رنگی قرمز، سبز و آبی می باشند و رنگ هر پیکسل در تصویر اصلی بر اساس شدت رنگ هر یک از این کانال‌ها تعیین می‌شود) و به صورت یک ماتریس سه بعدی ذخیره شده و نمایش داده می‌شوند.



در این ماتریس، ابعاد اول و دوم نشان‌دهنده‌ی طول و عرض تصویر و بعد سوم نشان‌دهنده‌ی شدت رنگ ۳ کانال رنگی می‌باشد. در واقع می‌توان هر پیکسل از تصویر را به صورت یک بردار با ۳ درایه نشان داد که درایه‌ی اول شدت رنگ قرمز، درایه‌ی دوم شدت رنگ سبز و درایه‌ی سوم شدت رنگ آبی را مشخص می‌کند.

$$pixel = \begin{bmatrix} Red\ value \\ Green\ value \\ Blue\ value \end{bmatrix}$$

در هنگام پیاده‌سازی، تصویر به صورت یک ماتریس با ابعاد (height, width, 3) به شما داده می‌شود و شما می‌توانید بسته به قابلیتی که در حال پیاده‌سازی آن هستید، از این ماتریس استفاده کنید.
در ادامه، فیلترهای مدنظر برای پیاده‌سازی شما توضیح داده شده‌اند.

سیاه-سفید کردن تصویر
در این قسمت باید ماتریس تبدیلی پیدا کنید که با ضرب کردن آن در تصویر ورودی، آن را سیاه سفید کند. می توانید برای اعمال این ماتریس تبدیل بر روی تصویر، از تابع utils.Filterاستفاده کنید.

بریدن تصویر
در این فیلتر سطر و ستون شروع و پایان برش داده می شود. تصویر جدید شامل پیکسل هایی که در بین این مقادیر قرار دارند، میشود.

تغییر اندازه‌ی تصویر (Scaling)
در این فیلتر، طول و عرض تصویر ورودی چند برابر می‌شود. برای تغییر اندازه تصویر، باید RGB تمامی پیکسل‌های تصویر با سایز جدید را بر حسب پیکسل‌های RGB تصویر قدیمی بنویسیم .برای اینکه بفهمیم دقیقا RGB کدام پیکسل قدیمی معادل RGB پیکسل در تصویر جدید است، از نسبت طول قدیم به طول جدید و عرض جدید به عرض قدیم برای طول و عرض هر پیکسل در تصویر جدید استفاده می‌کنیم.
توجه: دقت کنید که استفاده از توابع از پیش تعریف شده برای Scale کردن مجاز نخواهد بود.

فیلتر دلخواه
ابتدا یک ماتریس تبدیل دلخواه به انتخاب خودتان در نظر بگیرید و آن را روی تصویر اعمال کنید. سپس وارون ماتریس تبدیل خود را مجددا به تصویر خروجی مرحله قبل وارد کنید. خروجی نهایی باید با تقریب خوبی همان تصویر اولیه باشد. (برای به دست آوردن ماتریس وارون می‌توانید از تابع numpy.linalg.inv استفاده کنید.)

توضیحات پیاده‌سازی
در این پروژه شما باید قسمت هایی از کد main.py که با TODO مشخص شده‌اند را پیاده سازی کنید. به علاوه تعدادی از توابع مورد نیاز در کد utils.py برای شما به طور کامل پیاده سازی شده است و نیاز به تغییر ندارند.
در ادامه توضیح چند تابع مهم برایتان آورده شده است. دو تابع اول در فایل utils.py پیاده‌سازی شده‌اند و نیازی به تغییر ندارند.

تابع *ShowImage*

با استفاده از این تابع می توانید تصویر مورد نظرتان را به همراه عنوان در matplotlib نمایش دهید. همچنین در صورتی که مقدار save-file را True قرار دهید، تصویر خروجی در پوشه out ذخیره خواهد شد.

تابع *Filter*

با گرفتن تصویر مورد نظر و ماتریس تبدیل، تبدیل را برایتان انجام داده و تصویر خروجی را برمی‌گرداند. این تابع برای اعمال فیلتر به تصویر مورد نظر، بر روی تمامی پیکسل‌های آن پیمایش می‌کند و با اعمال ماتریس تبدیل به کانال‌های رنگی هر پیکسل، تصویر فیلتر شده را تولید می‌کند .ماتریس تبدیلی که به این تابع داده می‌شود، باید یک ماتریس 3 × 3 باشد تا مقدار سه کانال رنگی هر پیکسل را به 3 مقدار جدید این کانال‌های رنگی برای آن پیکسل، تبدیل کند.

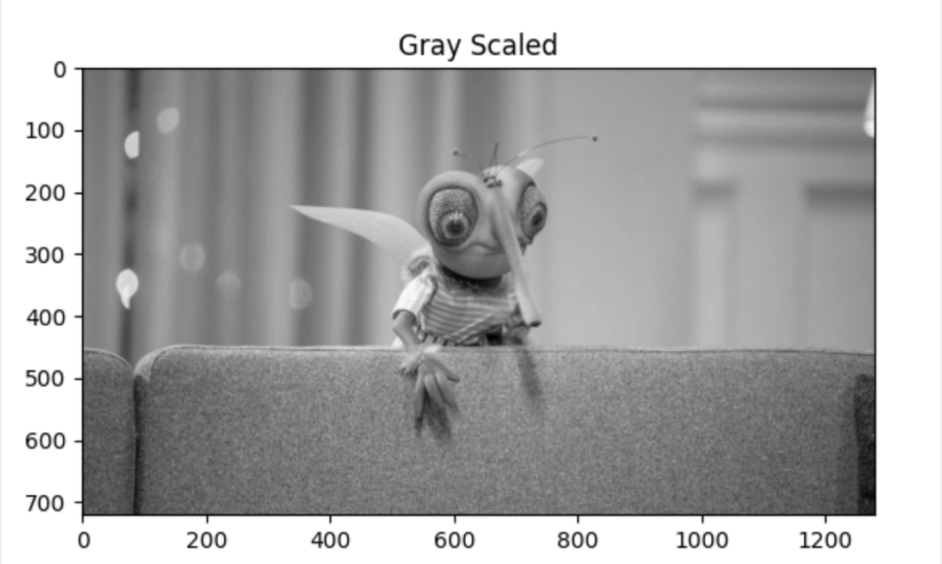
$$\begin{bmatrix} newRed \\ newGreen \\ newBlue \end{bmatrix} = M_{3\times 3} \begin{bmatrix} Red \\ Green \\ Blue \end{bmatrix}$$

برای مثال در صورتی که بخواهیم در یک تصویر، رنگ آبی را فیلتر کنیم، باید شدت کانال رنگ آبی را در تمامی پیکسل‌ها، صفر کنیم. برای این کار می توان از ماتریس تبدیل زیر استفاده کرد.

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

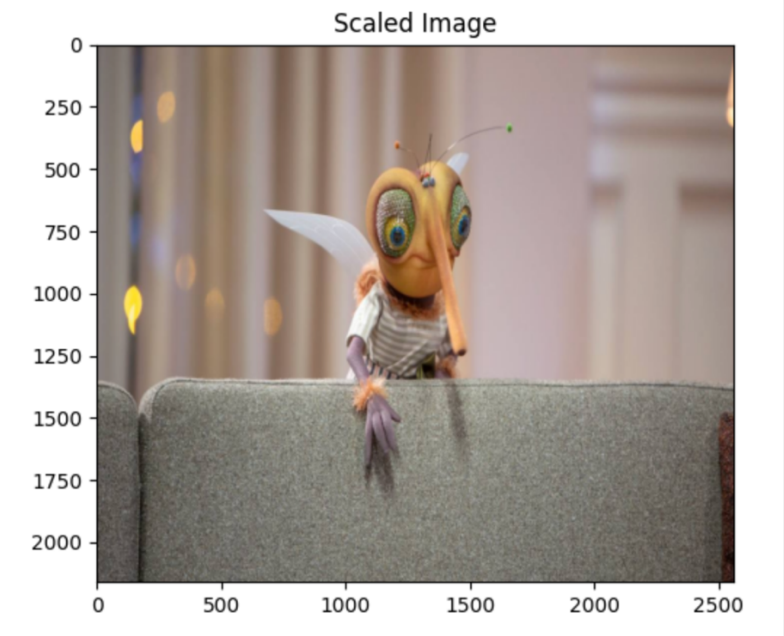
تابع grayScaledFilter

در این تابع شما نسخه‌ی رنگی تصویر را گرفته و با انجام عملیاتی، نسخه‌ی سیاه-سفید آن را برمی‌گردانید.



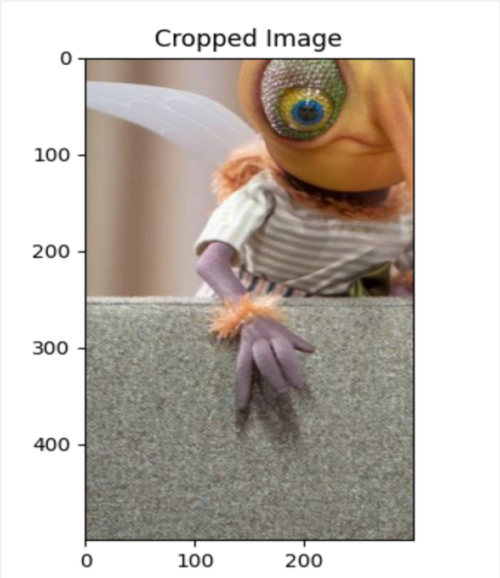
تابع scaleImg

در این تابع نیاز است که شما با توجه به طول و عرض جدیدی که به عنوان ورودی به این تابع داده می‌شود، تصویر جدید با این ابعاد را با اعمال عملیاتی بر روی تصویر اولیه به دست آورده و به عنوان خروجی برگردانید.



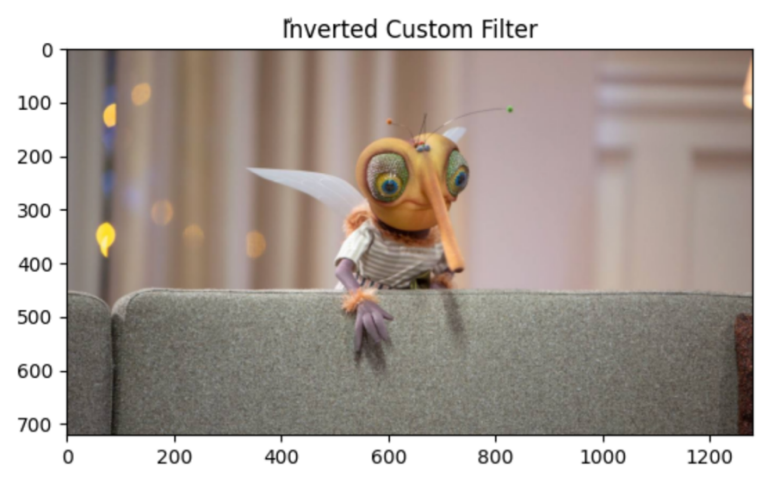
تابع cropImg

در این تابع نیاز است که شما با توجه به ابتدا و انتهای سطر و ستون هایی از تصویر که به دنبال آنها هستیم، این قسمت‌ها را از تصویر اصلی جدا کرده و به عنوان تصویر جدید در خروجی برگردانید.



تابع customFIlter

همانگونه که در قسمت "فیلتر دلخواه" توضیح داده شده است، این تابع را پیاده سازی کرده و تصاویر حاصل از اعمال فیلتر های مربوطه را، برگردانده یا مستقیما نمایش دهید.



نکات

- برای انجام این تکلیف تنها مجاز به پیاده‌سازی بخش‌های TODO در فایل main.py هستید. در نهایت این فایل را به همراه یک فایل pdf. با گزارش مختصری از آنچه انجام داده‌اید را به صورت zip شده در سامانه‌ی courses آپلود کنید.
- برای پیاده سازی این پروژه تنها مجاز به استفاده از زبان پایتون و کتابخانه Numpy در کنار توابع و کتابخانه‌های پیش فرض پایتون هستید .استفاده از هر زبان برنامه نویسی یا کتابخانه‌ای دیگر قابل قبول نبوده و در صورت استفاده، نمره‌ای به شما تعلق نخواهد گرفت.
- از رعایت تمیزی کد، استفاده از توابع مختلف برای پیاده سازی پروژه به شدت استقبال می شود.

موفق باشید