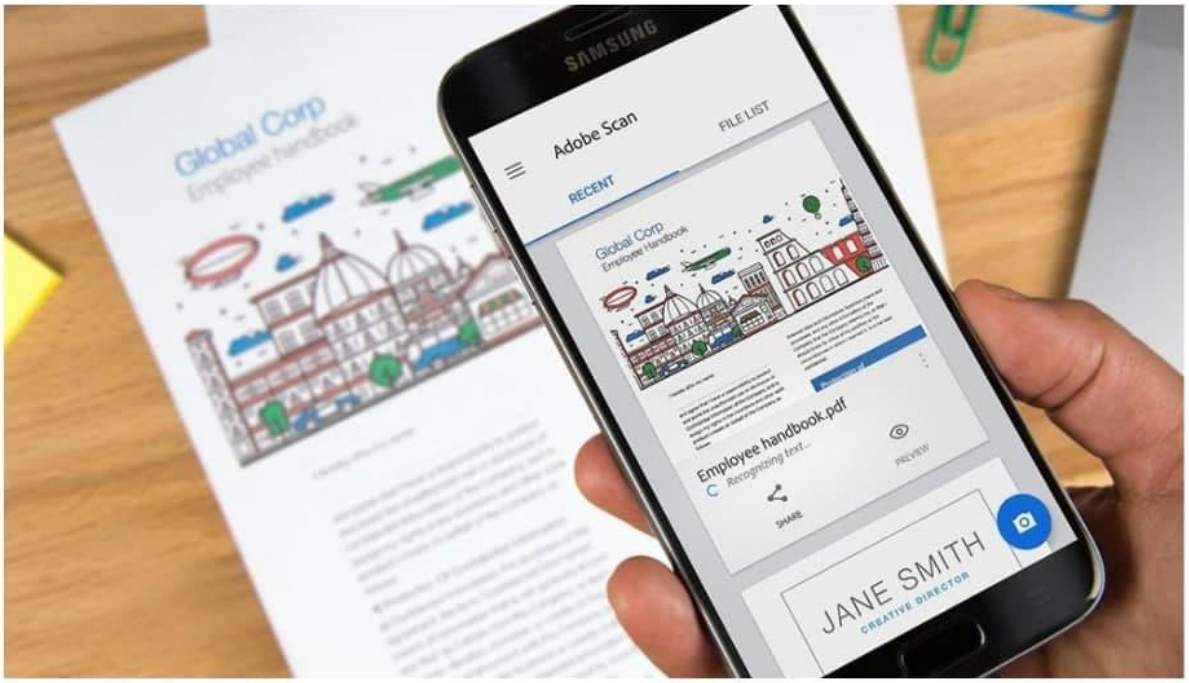
جبراسکنر

در این پروژه با استفاده از آنچه از تبدیل های خطی خوانده ایم، می‌خواهیم عملکرد برنامه‌های اسکن اسناد با موبایل (مثل CamScanner) را شبیه سازی کنیم.



فهرست مطالب

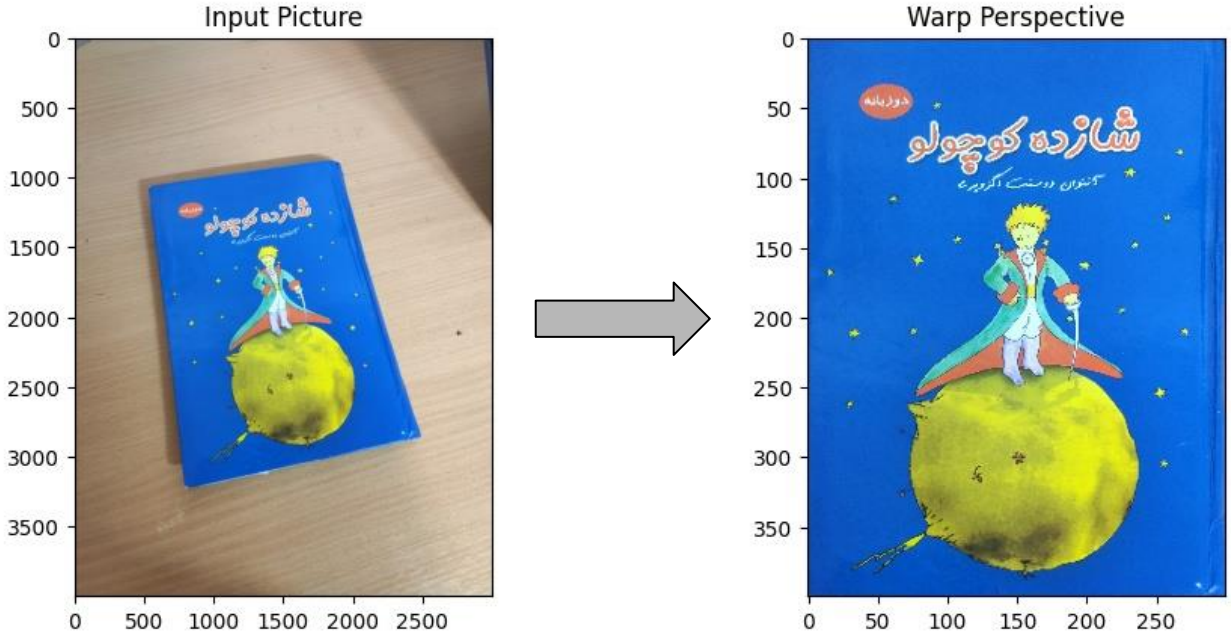
مقدمه

حتماً برایتان پیش آمده است که بخواهید از برگه کاغذی عکس بگیرید اما عکسی که گرفتید به طور صحیح گرفته نشده است و بخش هایی از آن خارج از قاب مستطیلی قرار می‌گیرد. با استفاده از برنامه‌های اسکن، به صورت خودکار تصاویرتان اصلاح شده و حتی می‌توانید روی آنها افکت های مختلف اعمال کنید و آنها را ویرایش کنید.

در این پروژه با طی کردن گام هایی که در ادامه بیان شده‌اند، میخواهیم جبراسکنر بسازیم :)

به صورت خلاصه، ابتدا مختصات چهار نقطه سندی که از آن عکس گرفته شده است را بدست می‌آوریم.

سپس یک ماتریس تبدیل پیدا می‌کنیم تا با اعمال آن بر روی چهار ضلعی حاصل بدست آمده از آن چهار نقطه، سند را به خوبی از حالت کج و معوج بودن در بیاورد.



در انتها نیز بعد از آنکه این سند به درستی از تصویر استخراج شد، فیلتر های مختلفی از جمله سیاه-سفید کردن، بریدن، بزرگ کردن و ... را بر روی آن اعمال می‌کنیم.

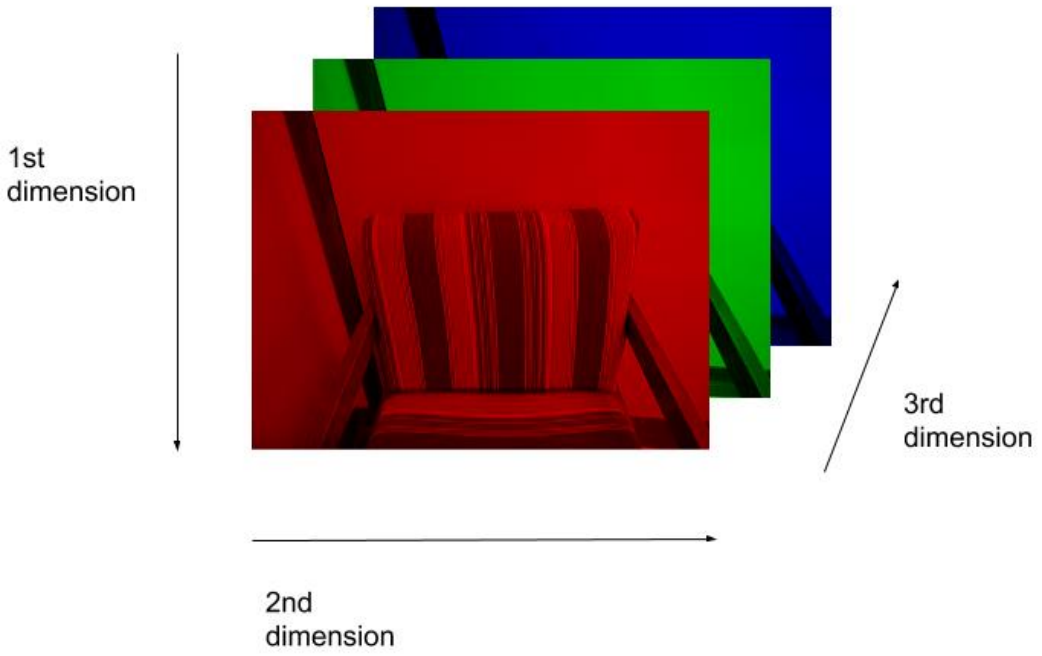
توضیح هر بخش به طور کامل برایتان آورده شده است.

امیدواریم با انجام این پروژه کاربردی، از جبرخطی لذت ببرین!

قبل از اینکه به سراغ گام های ساخت جبر اسکنر بریم، خوب است با ساختار تصاویر RGB آشنا شویم.

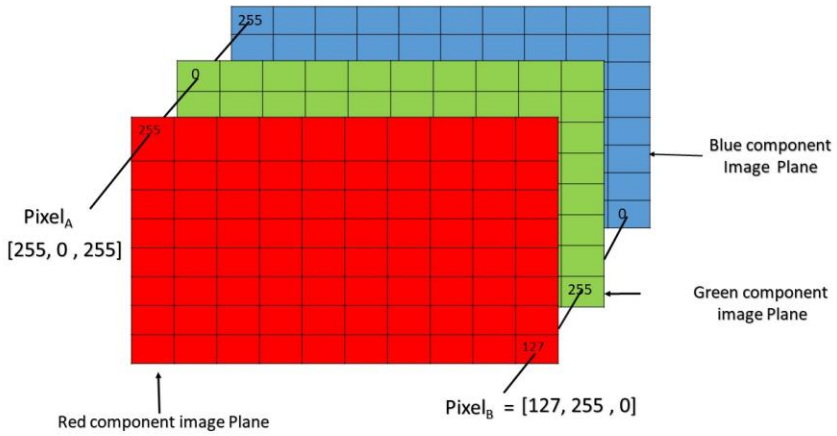
ساختار تصاویر RGB

در کامپیوتر ها، تصاویر رنگی از فرمت RGB (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 کانال ها تعیین می شود) می باشند و به صورت یک ماتریس سه بعدی ذخیره و نمایش داده می شوند.



همانطور که مشاهده می شود، بعد اول این ماتریس، ارتفاع تصویر، بعد دوم این ماتریس عرض تصویر و آخرین بعد، کانال های رنگی تصویر می باشد. در واقع می توان گفت در این حالت، ما به ازای هر پیکسل از تصویر یک بردار خواهیم داشت که اولین مقدار آن مربوط به شدت رنگ قرمز، دومین مقدار مربوط به شدت رنگ سبز و سومین مقدار مربوط به شدت رنگ آبی می باشد.

$$pixel_i = [Red\ value, Green\ value, Blue\ value]^T$$



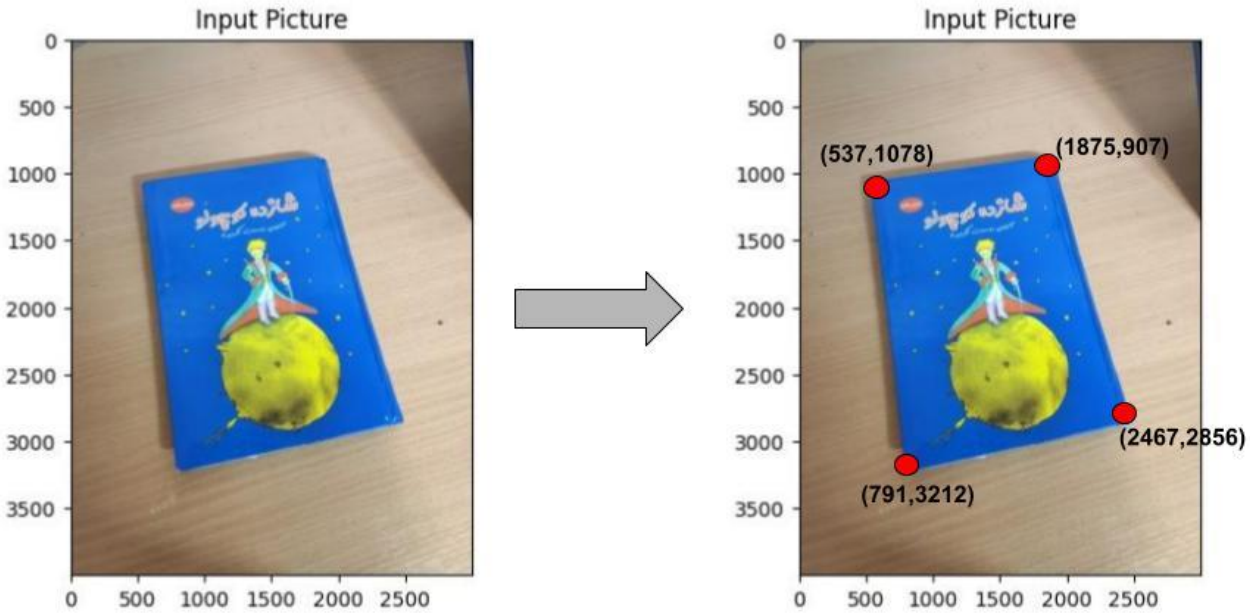
Pixel of an RGB image are formed from the corresponding pixel of the three component images

در هنگام پیاده سازی، زمانی که تصویر به صورت ماتریس به شما داده می شود، ابعاد آن به صورت $(height, width, 3)$ می باشد و شما می توانید بسته به قابلیتی که در حال پیاده سازی آن هستید، از این ماتریس استفاده کنید.

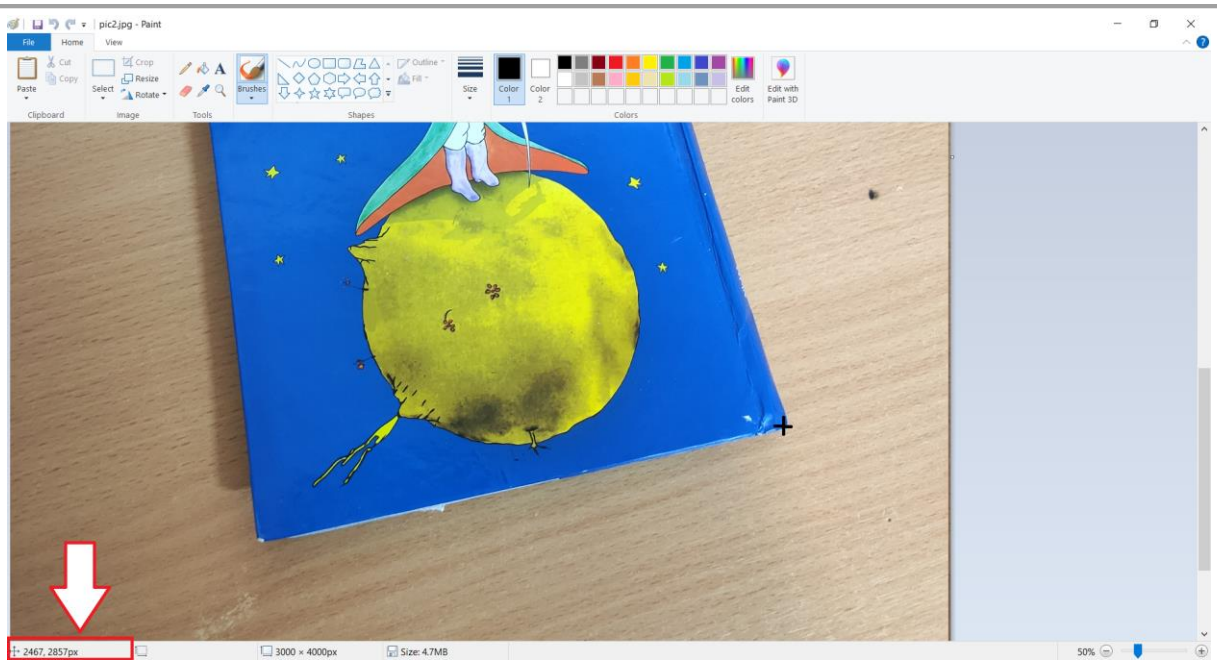
گام‌های ساخت جبراسکنر

گام اول) پیدا کردن چهار نقطه سند

برای پیدا کردن چهار نقطه سند، در این پروژه به صورت دستی با استفاده از ابزارهایی مثل Paint می‌توانیم مختصات چهار گوشه سند را پیدا کنیم و داخل یک آرایه ذخیره کرده و در کد قرار دهیم.



برای پیدا کردن مختصات پیکسلی چهارگوشه سند در تصویر تنها لازم است که پس از باز کردن تصویر در نرم افزار Paint، با تغییر موقعیت اشاره گر بر روی تصویر، مختصات آن نقطه که در نوار پایین سمت چپ در این نرم افزار نوشته می‌شود را پیدا کنید.



البته شما برای این کار محدود به نرم افزار Paint نیستید و می‌توانید از هر نرم افزاری که قابلیت مشابه این را داشته باشد، برای پیدا کردن مختصات این نقاط استفاده کنید.

گام دوم) پیدا کردن ماتریس تبدیل

برای اینکه عکسمان به درستی در یک قاب مستطیلی عمودی قرار گیرد، به یک ماتریس تبدیل نیاز داریم تا تصویر را اصلاح کند.

این ماتریس تبدیل به طور کامل پیاده سازی شده است و یک تابع کمکی برای این کار برای شما فراهم شده است که در فایل منابع همراه دستور کار، دراختیارتان قرار می‌گیرد.

در صورتی که علاقه‌مند بودید تا با چگونگی کارکرد این تابع آشنا شوید، می‌توانید از لینک زیر استفاده کنید.

Perspective projection, 4 points

گام سوم) اعمال ماتریس تبدیل

اکنون باید تصویر جدید را بسازیم و هر پیکسل را با استفاده از ماتریس تبدیل (T) به خروجی مورد نظرمان مپ کنیم.

فرض کنید می‌خواهیم نقطه (x, y) در تصویر اولیه را به خروجی مپ کنیم.

برای اینکار کافیست ماتریس تبدیل را بر روی بردار $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ اعمال کنیم:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = T_{3\times3} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

اکنون در فضای سه بعدی نقطه ی (x', y', z') را خواهیم داشت. اکنون باید این نقطه را به فضای ۲ بعدی ببریم و در واقع روی صفحه $z = 1$ ، تصویر کنیم:

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} \frac{x'}{z'} \\ \frac{y'}{z'} \end{bmatrix}$$

در نهایت ، مقدار پیکسل (x'', y'') در خروجی برابر است با مقدار پیکسل (x, y) در تصویر اولیه.

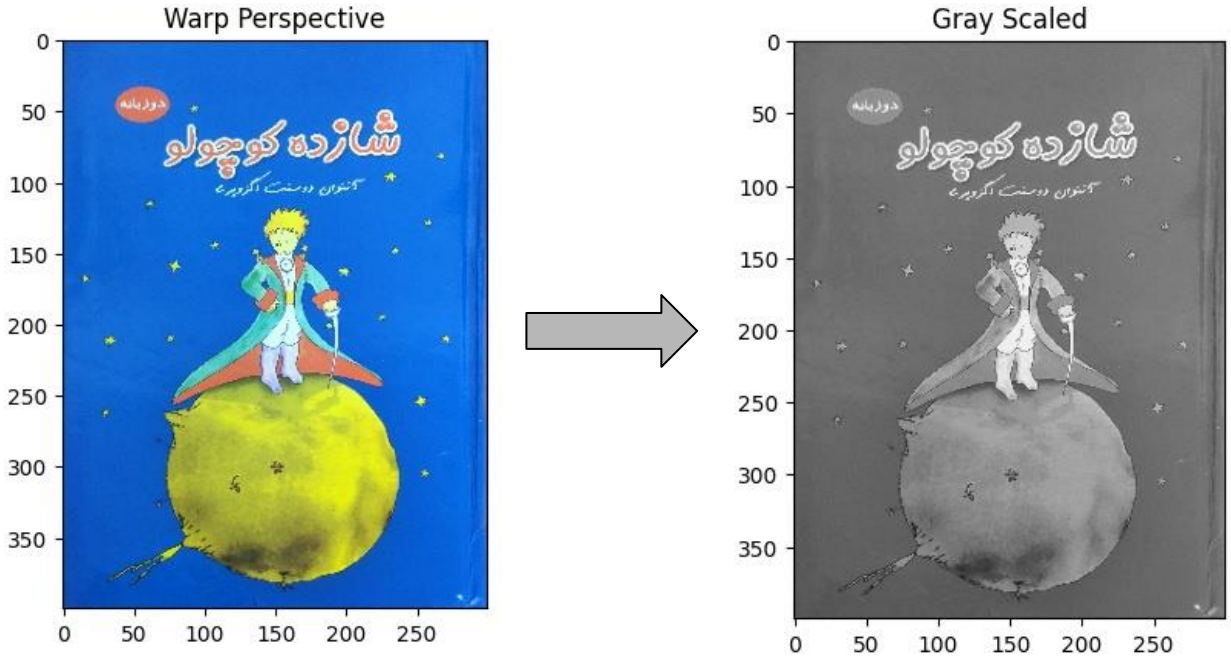
به همین صورت سعی می‌کنیم تمام پیکسل های تصویر اولیه را به خروجی مپ کنیم. (البته پیکسل‌هایی که بعد از مپ کردن، در محدوده ی ابعاد خروجی مد نظرمان باشد.)

گام چهارم) ویرایش سند استخراج شده (اعمال فیلتر)

در گام آخر، با اعمال فیلترهایی که در ادامه خواهیم دید، می‌خواهیم سند استخراج شده را ویرایش کنیم.

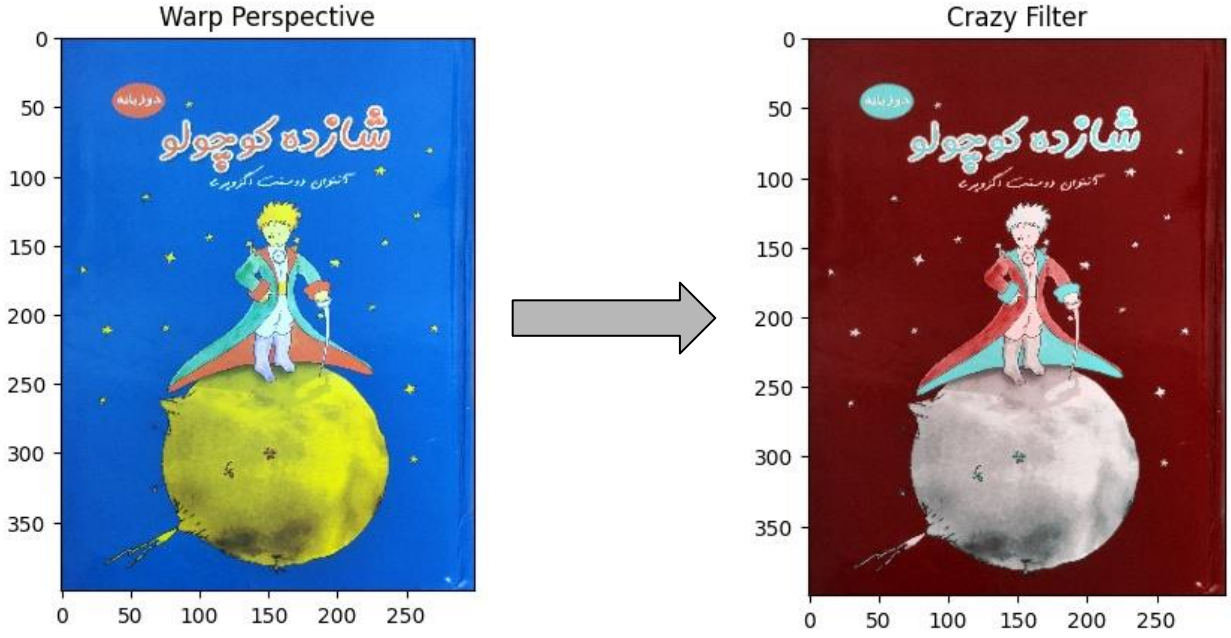
سیاه-سفید کردن تصویر

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.Filter استفاده کنید.



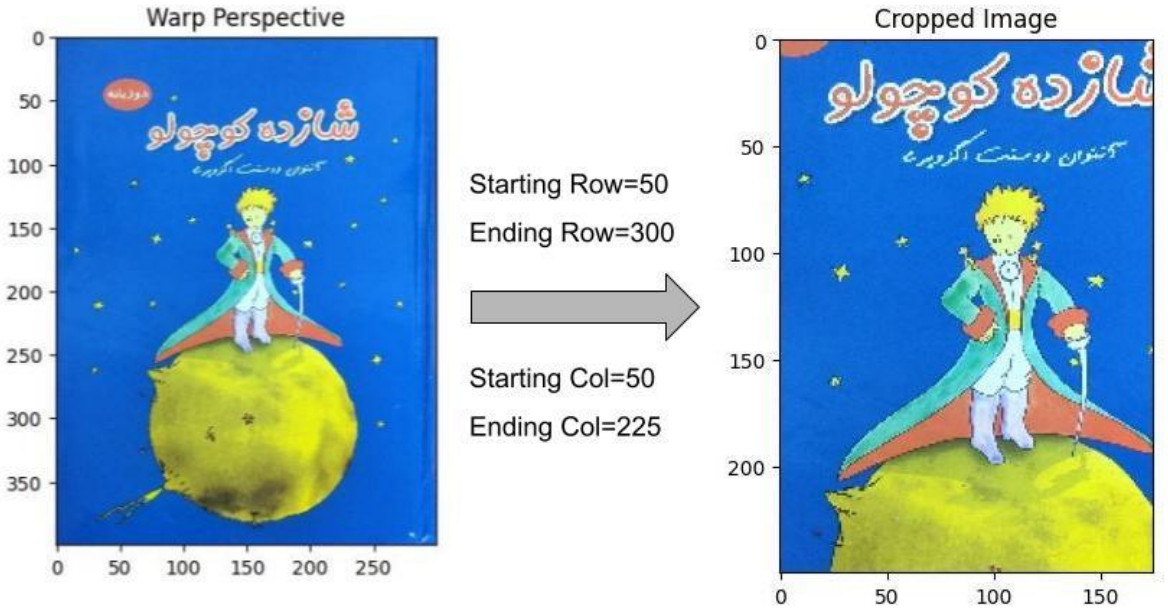
فیلتر دیوانه

در این فیلتر می‌خواهیم رنگ آبی را حذف کرده و رنگ سبز را در هر نقطه با رنگ قرمز جایگزین کنیم، همچنین می‌خواهیم رنگ قرمز را در هر نقطه برابر با مجموع رنگ آبی و سبز در آن نقطه قرار دهیم. بدین منظور، شما باید با استفاده از مفاهیمی که تا کنون یاد گرفته اید، ماتریس تبدیل مناسب برای این فیلتر را بیابید و آن را بر روی تصویر ورودی اعمال کنید.



بریدن تصویر

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



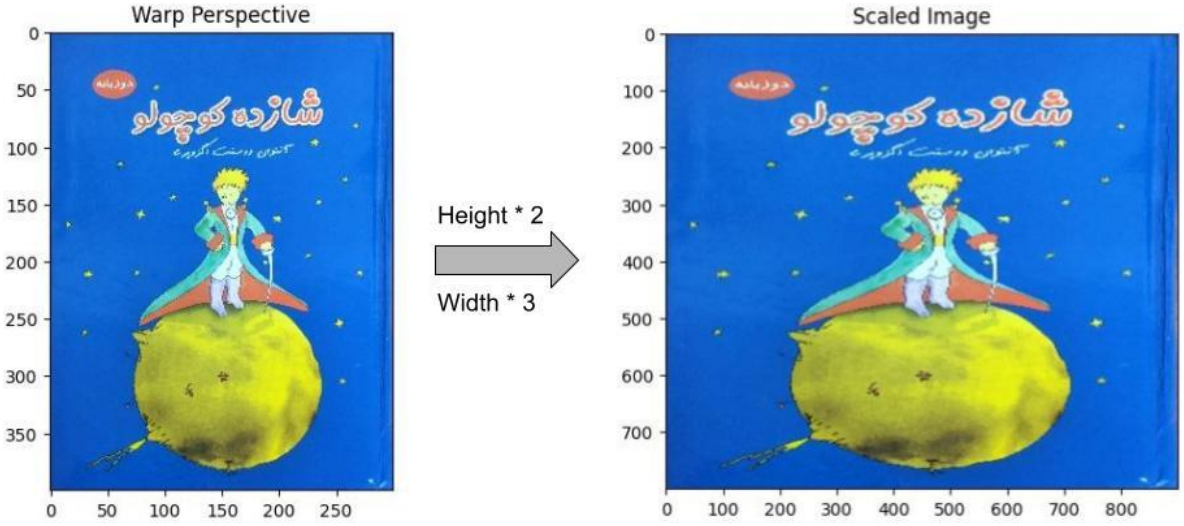
تغییر اندازه تصویر(Scaling)

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.

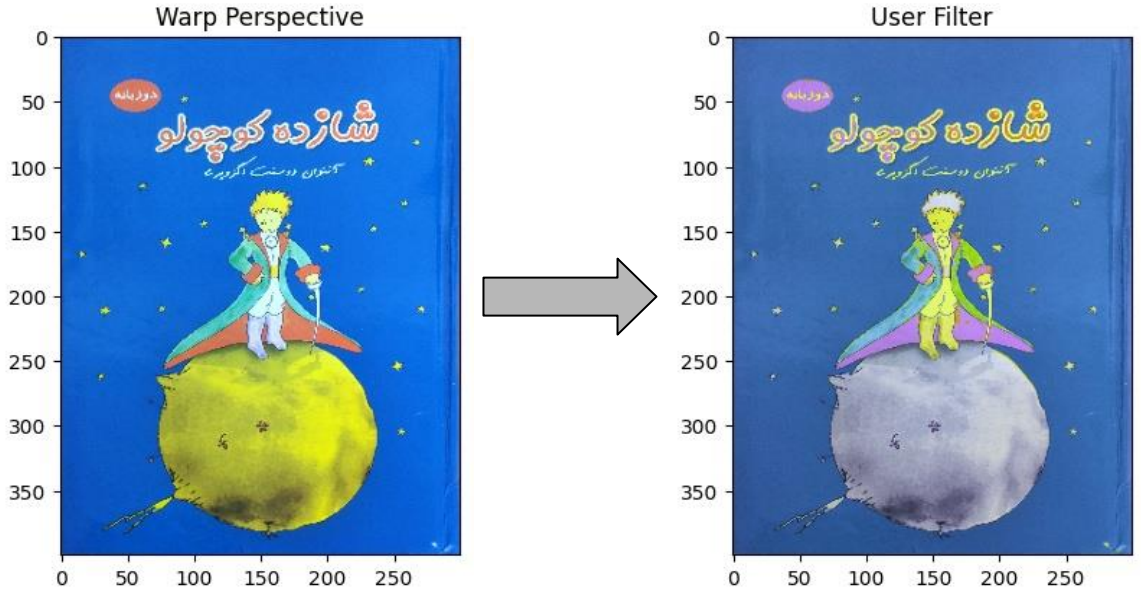
برای اینکه بفهمیم دقیقا RGB کدام پیکسل قدیمی معادل RGB پیکسل در تصویر جدید است، از نسبت طول قدیم به طول جدید و عرض جدید به عرض قدیم برای طول و عرض هر پیکسل در تصویر جدید استفاده می‌کنیم.

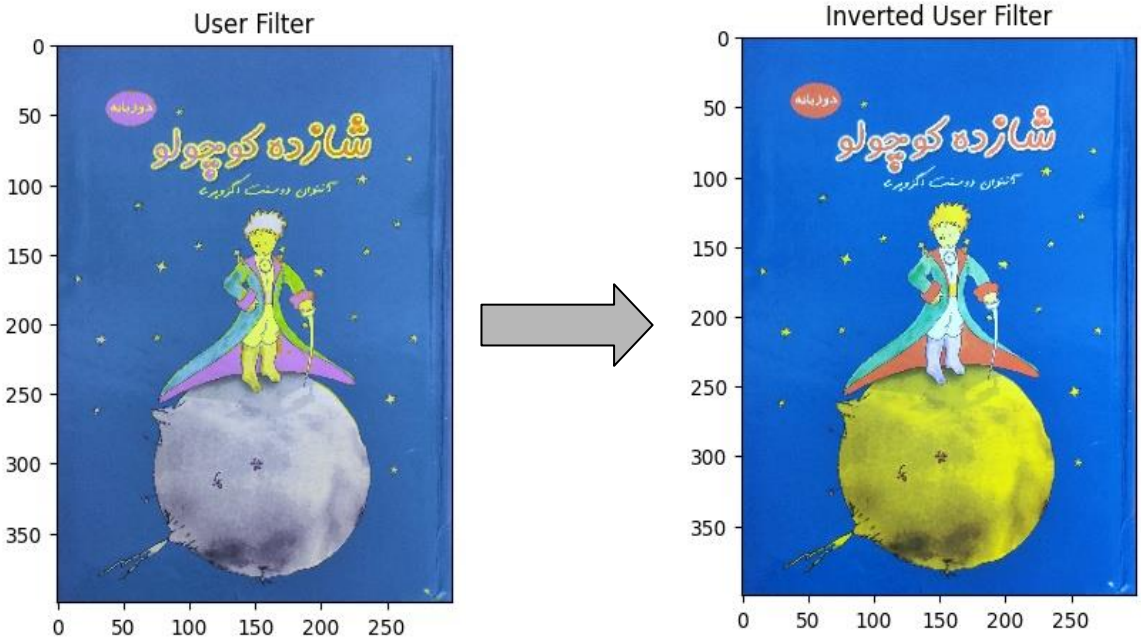
توجه: دقت کنید که استفاده از توابع از پیش تعریف شده برای Scale کردن مجاز نخواهد بود.



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 بدست 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 شده اند 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.
توابعی که به صورت ایتالیک نوشته شده‌اند (سه تابع اول)، از پیش پیاده‌سازی شده و در utils موجود هستند.

تابع *ShowImage*

با استفاده از این تابع می‌توانید 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 پیکسل های آن پیمایش می کند و با اعمال ماتریس تبدیل به کانال های رنگی هر پیکسل، تصویر فیلتر شده را تولید می کند.
ماتریس تبدیلی که به این تابع داده می شود، باید یک ماتریس 3×3 باشد تا مقدار سه کانال رنگی هر پیکسل را به ۳ مقدار جدید این کانال های رنگی برای آن پیکسل، تبدیل کنید.

$$\begin{bmatrix} newRed \\ newGreen \\ newBlue \end{bmatrix} = M_{3\times3} \begin{bmatrix} Red \\ Green \\ Blue \end{bmatrix}$$

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

$$M_{3\times3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

تابع *getPerspectiveTransform*

این تابع با گرفتن مختصات گوشه‌های سند در تصویر و نیز مختصاتی که قرار است به آن مپ شود، ماتریس تبدیل را می‌سازد.

تابع warpPerspective

در این تابع نیاز است که شما بر اساس توضیحات داده شده در قسمت "اعمال ماتریس تبدیل"، با داشتن ماتریس تبدیل، نحوه اعمال این ماتریس را پیاده سازی کنید و سپس ماتریس تصویر نهایی را برگردانید.

تابع grayScaledFilter

در این تابع شما تصویر رنگی (RGB) را دریافت کرده و با انجام عملیاتی نسخه سیاه-سفید (Gray Scale) این تصویر را به عنوان خروجی برمی‌گردانید.

تابع crazyFilter

در این تابع نیاز است که شما با توجه توضیحات موجود در قسمت "فیلتر دیوانه"، ماتریس تبدیلی پیدا کنید که این هدف را برآورده کند. سپس با اعمال آن بر تصویر اولیه، تصویر نهایی را به عنوان خروجی برگردانید.

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.
برای پیاده سازی این تابع، می توانید طبق توضیحی که در ادامه بیان می شود، کد را پیاده کنید.
فرض کنید میخواهیم مقدار نقطه ی (x, y) در خروجی را پیدا کنیم. برای x داریم:

$$newx = x \times \frac{oldWidth}{newWidth}$$

به همین صورت برای y داریم:

$$newy = y \times \frac{oldHeight}{newHeight}$$

بنابراین مقدار نقطه‌ی (x, y) در خروجی برابر است با مقدار خانه ی (newx, newy) در تصویر اولیه

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 قسمت ها 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.

قوانین و ددلاین

- ددلاین این پروژه، **ساعت ۲۳:۵۹ روز ۵ آذر** می باشد.

- تنها نیاز است کد main.py خود را بعد از تکمیل، در صفحه کوئرا آپلود کنید.

- هر دانشجو می‌بایست پروژه را به صورت انفرادی انجام دهد. تقلب‌ها به صورت خودکار، توسط سامانه کوئرا بررسی خواهد شد.

- از آن جایی که زبان برنامه نویسی پایتون، یکی از زبان های پرکاربرد در حوزه جبر خطی است و آموزش‌های مربوط به این زبان و کتابخانه‌های آن، توسط تیم تدریس‌یاری در اختیار شما قرار گرفته است، بنابراین 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 برنامه‌نویسی یا کتابخانه‌ای دیگر قابل قبول نبوده و در صورت استفاده، نمره‌ای به شما تعلق نخواهد گرفت.

- این پروژه تحویل آنلاین خواهد داشت و پس از پایان مهلت تحویل این پروژه، زمانبندی مربوط به تحویل به زودی اعلام خواهد شد.

- در صورتی که در رابطه با پروژه، سوالی برایتان پیش آمد، می توانید با یکی از تدریسیاران زیر سوال خود را در میان بگذارید.
  - @Amir_fal_01
  - @hawwwdi
  - @MSAJADCR7
  - @amirrezaRJ80
  - @AliAsad059

با آرزوی موفقیت و سلامتی

تیم تدریس یاری جبر خطی کاربردی، پاییز ۱۴۰۰