مسئولان تمرین: محمد فرهی، میثاق محقق

مقدمه

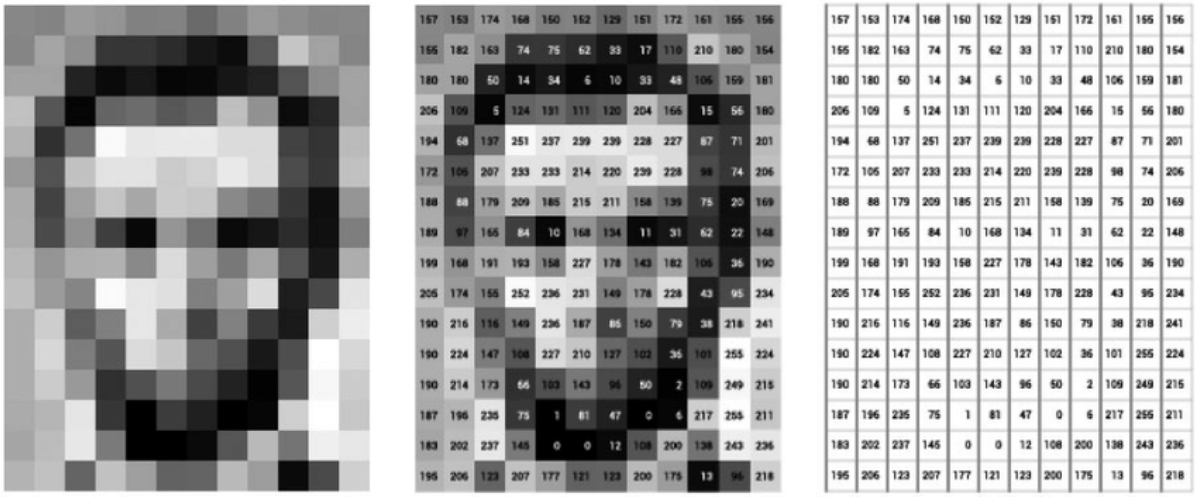
هدف از این تمرین آشنایی شما با مفاهیم اولیه طراحی چندریسه‌ای[1] یک مسئله است.
در این تمرین شما به اعمال فیلترهایی روی تصاویر می‌پردازید. این تصاویر در فرمت 24 بیتی بیت‌مپ (BMP) هستند و کد نحوه خواندن این تصاویر به شما داده شده و شما باید اعمال فیلترها روی این تصاویر را در دو حالت سریال و موازی پیاده‌سازی کنید.

شرح تمرین

در این تمرین شما پس از انجام مراحل اعمال فیلترهایی بر روی تصاویر، نتیجه که تصویری تغییر یافته است را به دست می‌آورید. در ابتدا برنامه شما اقدام به خواندن تصویر ورودی کرده و مقادیر سه کانال رنگی پیکسل‌های آن را در حافظه خود ذخیره می‌کند. پس از استخراج اطلاعات عکس، برنامه اقدام به اعمال مرحله به مرحله فیلترهای مورد نظر می‌کند. در این تمرین شما به دو روش سریال و موازی این مسئله را پیاده‌سازی می‌کنید.

خواندن تصویر

کد این قسمت در فایلی با نام bmp.cpp در کنار این پرونده به شما ارائه شده است و شما باید این کد را تکمیل کنید. برای آشنایی بیشتر با فرمت عکس، به این لینک مراجعه کنید. مقدار عددی هر پیکسل از تصویر در حالت RGB (مقادیر سه کانال رنگی قرمز، سبز و آبی) را باید در ساختمان داده دلخواه خود ذخیره کنید. از این مقادیر در مراحل بعدی برای ایجاد تغییر در تصویر استفاده خواهید کرد. همچنین این مقادیر در بازه 0 تا 255 هستند.



[1] Multi-Threaded Design

فیلترها

از عکس زیر جهت نشان دادن اثر فیلترها استفاده می‌کنیم:

فیلتر آینه عمودی

این فیلتر تصویر را به صورت عمودی آینه می‌کند. فیلتر از رابطه زیر پیروی می‌کند:

$$output(x,\ y)\ =\ input(x,\ \ -\ y)$$

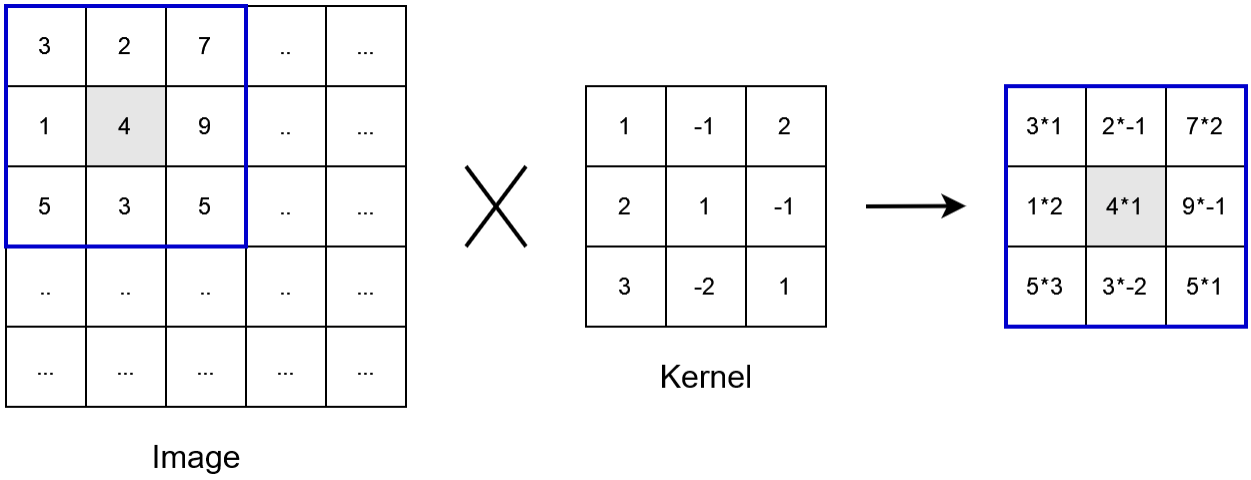
نتیجه این مرحله:

فیلتر با کرنل

این دسته از فیلترها به این صورت اند که هر پیکسل عکس ورودی و 8 پیکسل اطراف آن، در ماتریسی به نام کرنل ضرب شده تا مقدار جدید پیکسل به دست آید و جایگزین آن شود. می‌توانید برای فهم بهتر این موضوع، این لینک را مطالعه کنید.

در نهایت ماتریس کرنل در کل عکس کانولوشن می‌شود و به عکس نتیجه می‌رسیم. ضرب کرنل به ازای هر سه کانال رنگی RGB انجام می‌شود و مقدار نهایی هر کانال بین 0 و 255 محدود می‌شود (یعنی اگر مثلا کمتر از 0 شد، 0 در نظر می‌گیریم).

برای مثال، ضرب یک کرنل در پیکسل (1, 1) یکی از کانال‌های عکس را نشان می‌دهیم:

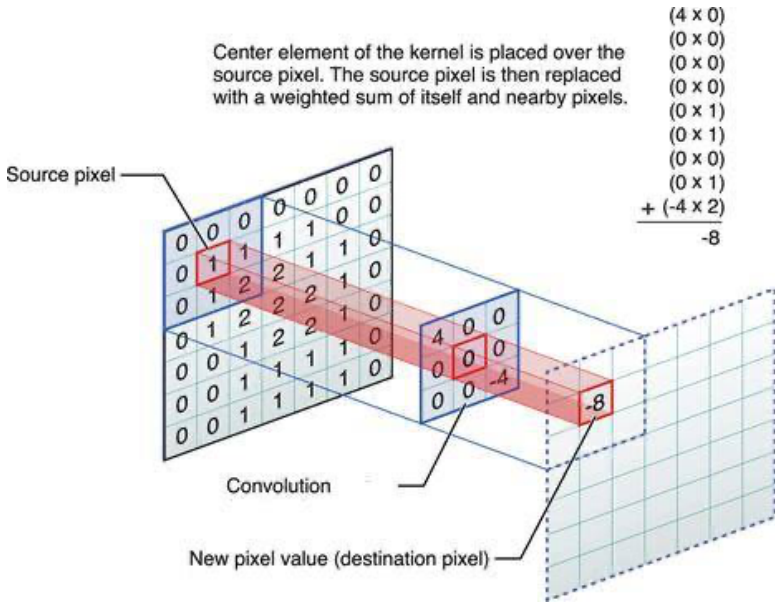


پس از ضرب، جمع حاصل 9 خانه (مقدار 26) به جای مقدار پیکسل خانه (1, 1) (مقدار اولیه 4) قرار می‌گیرد.

این کار به ازای همه پیکسل‌های عکس ورودی انجام می‌شود. توجه کنید که هنگام ضرب کرنل، باید از مقدار اولیه پیکسل‌ها استفاده شود و نه مقداری که قبلا طی اعمال فیلتر کرنل تغییر کرده است.

برای پیکسل‌هایی که چند تا از 8 پیکسل اطراف آنها از محدوده عکس خارج می‌شوند، می‌توانید هر روش edge handling دلخواهی را استفاده کنید که یکی از آنها، این است که مقدار خانه‌های خارج عکس را برابر مقدار خود خانه مرکزی در نظر بگیرید.

شکلی دیگر برای نشان دادن ضرب کرنل:



فیلتر تار کردن

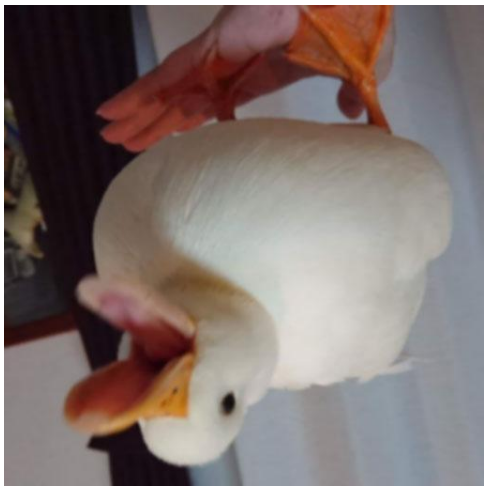
با کانولوشن کرنل زیر در عکس، فیلتر Gaussian Blur اعمال می‌شود که تصویر را تار می‌کند:

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Gaussian Blur Kernel

مقدار 1/16 در پشت کرنل، جهت normalize کردن ماتریس است و یعنی همه خانه‌های آن بر 16 (جمع مقدار خانه‌ها) تقسیم می‌شوند.

نتیجه این مرحله:



فیلتر رنگ Purple Haze

این فیلتر لایه‌ای بنفش به روی تصاویر اضافه می‌کند.

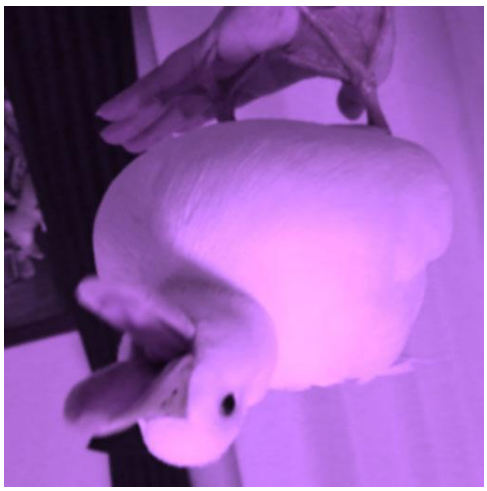
نحوه عملکرد این فیلتر به این صورت است که با کمک یک ماتریس ضرایب، تعیین می‌کند که چه مقدار از هر کانال رنگ در نتیجه نهایی حضور داشته باشد. برای ایجاد فیلتر Purple Haze می‌توانید از فرمول زیر برای هر پیکسل یک تصویر استفاده کنید:

$$New\ Red = 0.5\ Red + 0.3\ Green + 0.5\ Blue$$

$$New\ Green = 0.16\ Red + 0.5\ Green + 0.16\ Blue$$

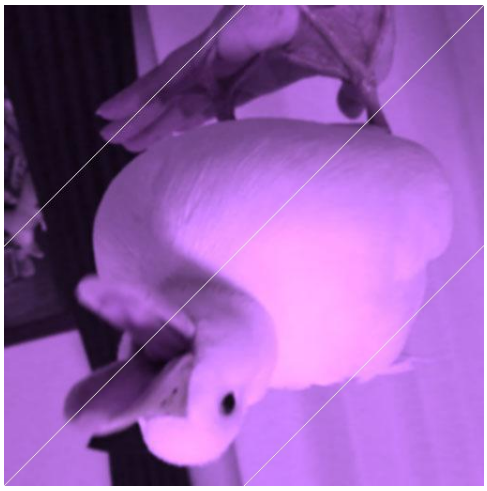
$$New\ Blue = 0.6\ Red + 0.2\ Green + 0.8\ Blue$$

نتیجه این مرحله:



افزودن هاشور مورب

در این مرحله، سه خط هاشور-مانند به تصویر اضافه می‌شوند.
نتیجه این مرحله به شکل زیر است:



پیاده‌سازی سری

در این بخش از تمرین شما به پیاده‌سازی serial برنامه خواسته شده می‌پردازید. سعی کنید در این بخش از تمرین، بهترین پیاده‌سازی را از لحاظ زمان اجرا انجام دهید؛ چرا که برای مقایسه عملکرد پیاده‌سازی چندریسه‌ای با سری، حالت سری باید در حالت بهینه پیاده‌سازی شده باشد. پس از این مرحله اعمالی که بیشترین زمان اجرا را به خود اختصاص داده‌اند را شناسایی کنید.

پیاده‌سازی چندریسه‌ای

در این بخش از تمرین، به موازی‌سازی اعمال صورت گرفته در توابعی که در بخش قبل به عنوان Hotspot (توابعی که بیشترین زمان زمان اجراها را به خود اختصاص می‌دهند) از آنها یاد کردید می‌پردازید.
خواندن ورودی و ذخیره‌سازی آن در حافظه از اعمال زمان‌گیر در بسیاری از برنامه‌هاست که احتمالا از توابع مربوط به آنها (در کنار سایر توابع) به عنوان Hotspot-های برنامه یاد کرده‌اید. برای موازی‌سازی این بخش می‌توانید خواندن و ذخیره‌سازی مقادیر پیکسل‌های تصویر و اعمال فیلتر روی آنها را توسط چندین ریسه انجام دهید. بهترین ترکیب تعداد ریسه‌ها، نحوه تقسیم داده‌ها و مکانیزم‌های همگام‌سازی ریسه‌ها را باید بدست آورده و انتخاب های خود را توجیه کنید. در انتها، میزان تسریع پیاده‌سازی چندریسه‌ای به پیاده‌سازی سری را از رابطه زیر بدست آورده و طبق قالب خروجی که در ادامه آمده است، گزارش کنید:

$$speedup = \frac{serial\ execution\ time}{parallel\ execution\ time}$$

- دقت کنید که خروجی برنامه چندریسه‌ای شما باید مطابق با خروجی برنامه سری شما باشد.
- توجه شود که این بخش از تمرین باید به صورت چندریسه‌ای پیاده‌سازی گردد و سایر پیاده‌سازی‌ها قابل قبول نیست.
- دقت شود برای موازی‌سازی پروژه، تنها مجاز به استفاده از کتابخانه pthreads هستید و استفاده از کتابخانه‌های دیگر (به جز کتابخانه‌های پایه زبان C++) مجاز نیست.

ورودی و خروجی برنامه

برنامه شما باید نام فایل تصویر ورودی را از خط فرمان دریافت کند. نمونه اجرای برنامه با فرض اینکه تصویر ورودی با نام input.bmp در کنار فایل اجرایی شما قرار گرفته است در زیر آمده است:

```
./ImageFilters.out input.bmp
```

برای هر دو پیاده‌سازی سری و چندریسه‌ای، باید عکس خروجی در فایلی به نام output.bmp ذخیره شده و زمان اجرای خواندن عکس، زمان اجرای هر فیلتر و کل زمان طی شده در برنامه پس از اجرا چاپ شود. یک نمونه خروجی برنامه در زیر آمده است:

```
Read: 4.751 ms
Flip: 2.513 ms
Blur: 62.896 ms
Purple: 6.483 ms
Lines: 0.127 ms
Execution: 78.168 ms
```

نکات تکمیلی

- تمام خروجی‌های برنامه را در جریان خروجی استاندارد (stdout) چاپ کنید و فقط عکس نتیجه به عنوان فایلی جدا تولید می‌شود.
- تضمین می‌شود که ورودی‌هایی که به برنامه شما داده می‌شود صحیح هستند و نیازی به بررسی صحت ورودی توسط برنامه شما نیست.
- طراحی درست، کارایی[2] برنامه و شکستن برنامه به بخش‌های کوچکتر تأثیر زیادی در نمره تمرین دارد.

نکات تحویل

- دقت کنید که فایل آپلودی شما با نام **OS_CA3_<SID>.zip** حتما باید شامل **دو پوشه** مجزا باشد که در یک پوشه پیاده‌سازی سری **(پوشه serial)** و در پوشه دیگر پیاده‌سازی موازی **(پوشه parallel)** آورده شده است.
- دقت کنید که فایل zip شما شامل فولدر بیرونی نباشد و مستقیماً پس از unzip کردن آن، دو پوشه ذکر شده پیاده‌سازی سریال و موازی شما بدست آید.
- تصویر ورودی و خروجی را در فایل آپلودی خود قرار ندهید.

برای مثال، یک نمونه فایل آپلودی مورد قبول در زیر آمده است:

```
OS_CA3_81019xxxx.zip
├── parallel
|   ├── main.cpp
|   └── makefile
└── serial
    ├── main.cpp
    └── makefile
```

- برنامه شما باید در سیستم‌عامل لینوکس و با مترجم g++ با استاندارد C++ 11 ترجمه و در زمان معقول برای ورودی‌های آزمون اجرا شود.
- دقت کنید که پروژه شما باید دارای Makefile باشد. همچنین در Makefile خود مشخص کنید که از استاندارد C++ 11 استفاده می‌کنید.
- نام فایل اجرایی شما که در کنار Makefile خود ساخته می‌شود باید **ImageFilters.out** باشد.
- نکته‌هایی که در جلسه توجیهی تمرین گفته می‌شود و یا در فروم‌های مربوطه مطرح می‌شوند بخشی از تمرین هستند؛ بنابراین به آنها توجه داشته باشید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.

[2] Performance