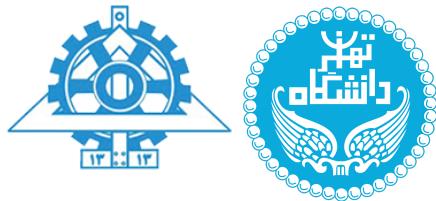


به نام خدا

تمرین کامپیوتری سوم



دانشکده مهندسی برق و کامپیوتر

سیستم‌های عامل - بهار ۱۴۰۳

مسئولان تمرین : [یهاد علمی](#) - [یوربا تاج محرب](#)

مهلت تحويل :

ساعت ۲۳:۵۹ روز ۲۹ اردیبهشت

مقدمه

هدف از این تمرین آشنایی شما با مفاهیم اولیه طراحی چندریسه‌ای^۱ یک مسئله است. در این تمرین شما به اعمال فیلترهایی روی تصاویر می‌پردازید.

این تصاویر در فرمت 24 بیتی بیتمپ^۲ 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 به اعمال فیلتر روی تصاویر می‌پردازید و پس از انجام مراحل، نتیجه، که تصویری تغییر یافته است به دست می‌آید. در بخش اول برنامه شما اقدام به خواندن تصاویر ورودی کرده و مقادیر سه کanal رنگی پیکسل های آن را در حافظه خود ذخیره می‌کند. در

بخش دوم برنامه شما از تصویر به صورت طیف خاکستری^۳ استفاده می‌کند تا در نهایت عملیات تشخیص لبه‌ها^۴ انجام شود.

در هر بخش برنامه شما فیلترهای مرتبط را مرحله به مرحله اعمال می‌کند. در این تمرین شما موظفید دو روش سری و موازی را پیاده‌سازی کرده و میزان تسریع نسخه چندنخی^۵ نسبت به نسخه سری را بسنجدید.

خواندن و پردازش تصاویر با استفاده از OpenCV

برای خواندن تصاویر از کتابخانه OpenCV استفاده خواهید کرد. این کتابخانه به شما کمک می‌کند تا درگیر جزئیات پیاده‌سازی خواندن فایل تصویر و ذخیره‌سازی آن نشوید و بتوانید راحت‌تر عملیات مورد نظر را روی تصاویر خود انجام دهید. برای نصب OpenCV می‌توانید از

¹ Multi-Threaded Design

² Bitmap

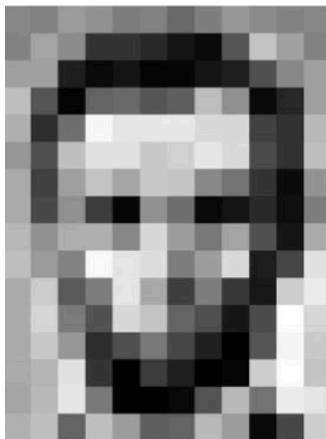
³ Grayscale

⁴ Edge Detection

⁵ Multi-Threaded

این [لینک](#) استفاده کنید.

توجه کنید که پیاده‌سازی فیلترها با استفاده از توابع آماده OpenCV مجاز نیست و باید خودتان آن‌ها را پیاده‌سازی کنید.



157	153	174	168	160	162	125	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	54	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	105	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	216	127	103	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	6	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	160	162	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	54	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	105	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	216	127	103	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	6	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

تصویر ورودی

تصویر زیر به شما داده شده است



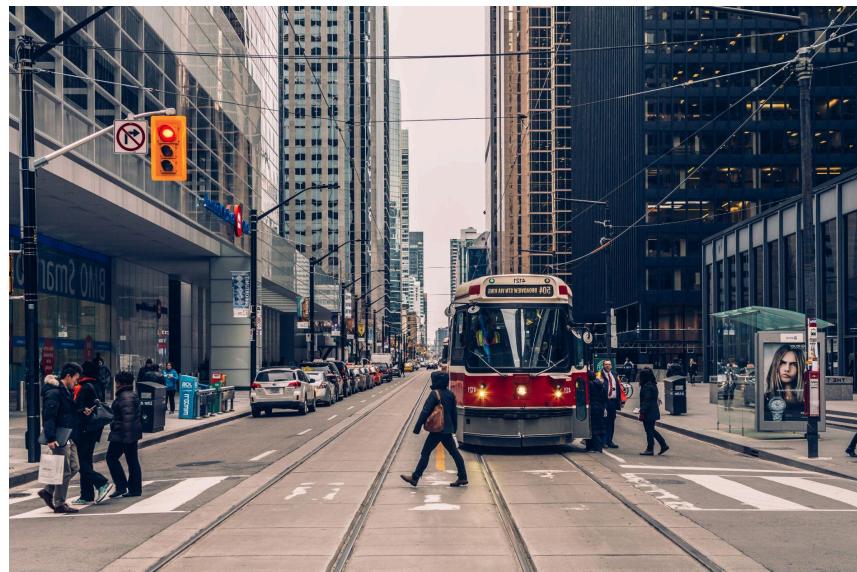
بخش اول: اعمال فیلتر بر تصویر رنگی

در این بخش به اعمال فیلترهای ذکر شده بر تصویر رنگی خواهید پرداخت، به این صورت که خروجی هر فیلتر، ورودی برای فیلتر بعدی خواهد بود.

فیلتر آیینه افقی

این فیلتر، تصویر را به صورت آیینه می‌کند. برای اعمال این فیلتر از رابطه زیر استفاده کنید.

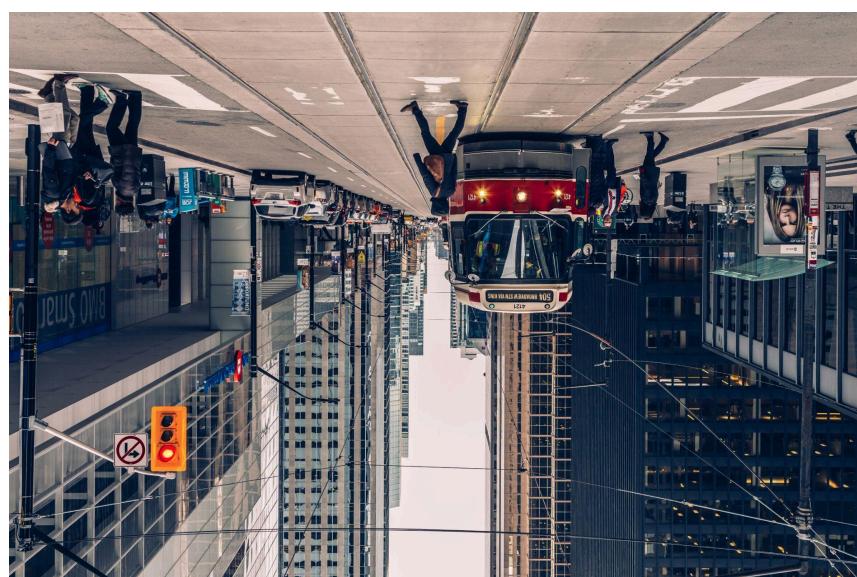
$$outputImage(x, y) = inputImage(-x, y)$$



فیلتر آیینه عمودی

این فیلتر، تصویر را به صورت آیینه عمودی می‌کند. برای اعمال این فیلتر از رابطه زیر استفاده کنید.

$$outputImage(x, y) = inputImage(x, -y)$$



اعمال فیلتر سپیا^۶

این فیلتر به تصاویر حس گرم و قدیمی بودن می‌دهد. این فیلتر با افزایش سطح رنگ‌های قرمز و زرد و کاهش سطح رنگ‌های آبی و سبز، به عکس یک رنگ قهوه‌ای-قرمز می‌بخشد که برای نوستالژیک ساختن عکس‌ها مورد استفاده قرار می‌گیرد. این فیلتر به این صورت عمل می‌کند که با کمک یک ماتریس ضرایب تعیین می‌کند که چه مقدار از هر کanal رنگ (قرمز، سبز و آبی) در نتیجه نهایی حضور داشته باشد. برای ایجاد فیلتر سپیا می‌توانید از فرمول زیر برای هر پیکسل یک تصویر استفاده کنید.

$$TR = 0.393R + 0.769G + 0.189B$$

$$TG = 0.349R + 0.686G + 0.168B$$

$$TB = 0.272R + 0.534G + 0.131B$$



بخش دوم: تشخیص لبه بر تصویر طیف خاکستری

یکی از کاربردهای تشخیص لبه‌ها در صنعت خودروسازی برای خودروهای خودران است. تشخیص لبه‌ها به خودروها کمک می‌کند تا درک بهتری از محیط اطراف (خط کشی‌ها، علائم راهنمایی و رانندگی، افراد، موائع و ...) داشته باشند. در این بخش یکی از راههایی که برای تشخیص لبه‌ها استفاده می‌شود با استفاده از عملیات ذکر شده پیاده‌سازی می‌کنید. در این بخش، ورودی شما تصویر طیف خاکستری می‌باشد. برای به دست آوردن تصویر طیف خاکستری می‌توانید از تصویر ابتدایی یک کپی عمیق⁷ تهیه کنید و با استفاده از توابعی که OpenCV در اختیاراتان می‌گذارد آن را تبدیل به تصویر طیف خاکستری کنید و یا تصویر را بار دیگر ولی این بار در حالت طیف خاکستری بخوانید.

⁶ Sepia Filter

⁷ Deep Copy

فیلتر Box Blur

تار کردن^۸ یکی از مراحلی است که پیش از انجام تشخیص لبه بر عکس‌ها انجام می‌شود. این کار باعث افزایش یکپارچگی و کاهش نویز تصویر می‌شود تا عمل تشخیص لبه بهتر انجام گیرد.
این فیلتر حاصل میانگین همسایه‌های یک پیکسل خواهد بود. در این قسمت یک فیلتر ۳ در ۳ از Box Blur (به تعبیر دیگر تا یک ردیف و ستون در میانگین‌گیری شرکت می‌کنند) را پیاده‌سازی می‌کنید. توجه کنید که میانگین‌گیری ساده کفایت می‌کند و اعمال الگوریتم‌های بهینه‌سازی میانگین‌گیری در ماتریس لازم نیست.
مثال زیر را در نظر بگیرید که عملیات برای عنصر وسط ماتریس اتفاق می‌افتد.

$$\begin{bmatrix} 0 & 2 & 3 \\ 5 & 1 & 6 \\ 7 & 6 & 0 \end{bmatrix} \xrightarrow{\text{Box Blur}} \begin{bmatrix} 0 & 2 & 3 \\ 5 & 3 & 6 \\ 7 & 6 & 0 \end{bmatrix}$$
$$= \frac{1}{9}(0 + 2 + 3 + 5 + 1 + 6 + 7 + 6 + 0) = 3$$



تبديل طيف خاڪستري به سياه و سفيد

پس از تار کردن تصویر، قدم بعدی به دست آوردن تصویر سیاه و سفید می‌باشد. برای این منظور از آستانه‌گذاري^۹ استفاده می‌شود. در این قسمت از آستانه‌گذاري دودوبي^{۱۰} با آستانه 127 استفاده خواهيد کرد که فرمول آن در زير آمده است.

⁸ Blur

⁹ Thresholding

¹⁰ Binary Thresholding

$$dst(x, y) = \begin{cases} 255 & : src(x, y) > 127 \\ 0 & : otherwise \end{cases}$$

خروجی این مرحله:



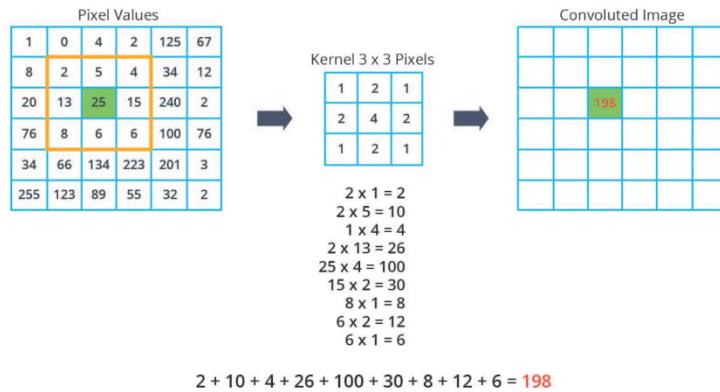
Laplacian Edge Detection

در این قسمت به پیاده‌سازی فیلتری می‌پردازید که لبه اشیا را تشخیص می‌دهد. برای این منظور از کرنل زیر استفاده کرده و هر کدام از پیکسل‌های تصویر که واجد شرایط هستند، روی آن‌ها عملیات Convolution با این کرنل انجام داده و مقدار حاصل را جایگزین کنید. دقت کنید که برای انجام Convolution علاوه بر مقدار خود پیکسل نیاز به مقادیر ۸ پیکسل اطراف آن هم دارید.

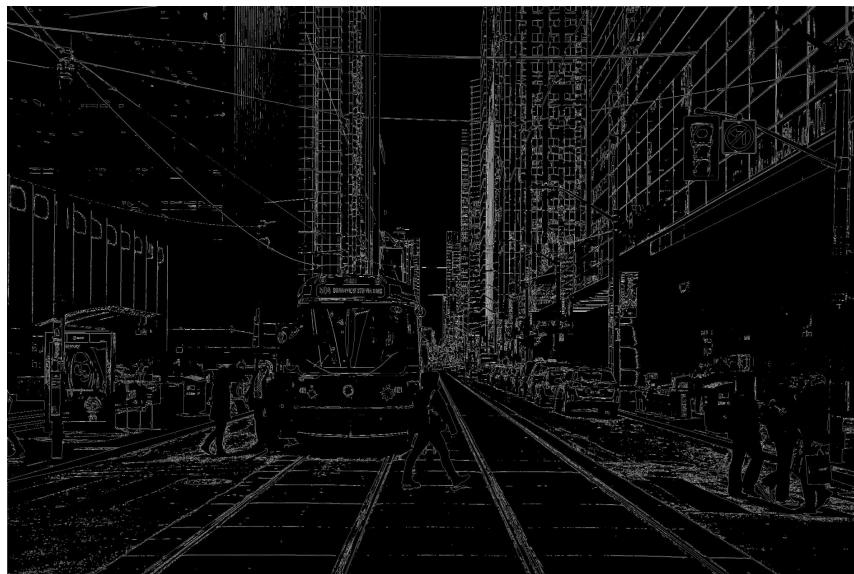
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

کرنل مورد استفاده در Laplacian Edge Detection

برای نحوه انجام Convolution می‌توانید از تصویر زیر کمک بگیرید



خروجی تصویر پس از تار کردن، تبدیل به سیاه و سفید و در نهایت اعمال فیلتر لابلاس، به شکل زیر خواهد بود:



پیاده‌سازی سری

در این بخش از تمرین شما به پیاده‌سازی سری 6 برنامه خواسته شده می‌پردازید. سعی کنید در این بخش از تمرین بهترین پیاده‌سازی از لحظه زمان اجرایی انجام دهید؛

زیرا برای مقایسه عملکرد پیاده‌سازی چند ریسنهای 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 (Hotspots) شناسایی کنید و در هنگام تحويل به صورت شفاهی توضیح دهید.

پیاده‌سازی چندریشه‌ای

در این بخش از تمرین به موازی‌سازی اعمال صورت گرفته در توابعی که در بخش قبیل به عنوان Hotspot از آنها یاد کردید می‌پردازید. شما بهترین ترکیب تعداد ریسه‌ها، نحوه تقسیم داده‌ها و مکانیزم‌های همگام‌سازی ریسه‌ها را باید به دست آورده و انتخاب‌های خود را توجیه کنید. در انتهای، میزان تسریع پیاده‌سازی چندریشه‌ای به پیاده‌سازی سری را از رابطه زیر بدست آورده و با توجه به خروجی و قالب آن که در ادامه آمده است در هنگام تحويل گزارش کنید:

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

نکات

- دقت کنید که خروجی برنامه چند ریشه‌ای شما باید با برنامه سری شما برابر باشد
- توجه شود که این بخش از تمرین باید به صورت چند ریسه ای پیاده سازی گردد و سایز پیاده سازی ها قابل قبول نیست
- دقت شود که برای موازی سازی پروژه فقط مجاز به استفاده از کتابخانه **PThread** هستید و استفاده از کتابخانه های دیگر برای موازی سازی مجاز نیست
- نام فایل اجرایی شما در هر دو حالت سری و موازی باید `ImageFilters.out` باشد

ورودی و خروجی برنامه

در انتهای می‌باشد خروجی نهایی هر یک از بخش‌های صورت پروژه خود ذخیره کنید. برای این کار می‌توانید از OpenCV استفاده کنید. برای نسخه سری از نام‌های `First_Serial.bmp` و `Second_Serial.bmp` و برای بخش موازی از نام‌های `First_Parallel.bmp` و `Second_Parallel.bmp` استفاده کنید. برنامه شما باید نام تصویر ورودی را از خط فرمان دریافت کند. نمونه اجرای برنامه با فرض اینکه تصویر ورودی با `ut.bmp` در کنار فایل اجرایی شما قرار گرفته است در زیر آمده است. خروجی گفته شده باید برای هر دو پیاده‌سازی سری و چندریشه‌ای، بعد از اجرای **هرکدام** به عنوان خروجی چاپ شود.

نمونه اجرا

```
./ImageFilters.out ut.bmp
```

قالب خروجی

```
Execution Time : <execution_time>
```

نمونه خروجی

```
Execution Time : 2.12
```

توجه: نام فایل خروجی برنامه و متن قالب خروجی را از اینجا کپی نکنید و آن را تایپ کنید!

نکات تكميلی

- تمام خروجی‌های متنی برنامه را در جریان خروجی استاندارد¹¹ چاپ کنید.
- تضمین می‌شود که ورودی‌هایی که به برنامه شما داده می‌شود صحیح هستند و نیازی به بررسی صحت ورودی توسط شما نیست
- طراحی درست، کارایی برنامه و شکستن برنامه به بخش‌های متناسب تاثیر زیادی در نمره‌دهی دارد

نحوه تحويل

- دقت کنید که نام فایل آپلودی، شما به نام OS_CAS_<SID>.zip حتماً باید شامل دو پوشه مجزا باشد که در یک پوشه پیاوه سازی سری و در پوشه دیگر پیاوه سازی موازی آورده شده است. دقت کنید که فایل zip شما شامل فولدر بیرونی نباشد و مستقیماً پس از unzip کردن آن، دو پوشه پیاوه سازی سری و موازی به دست آید. تصویرهای ورودی و خروجی را در فایل آپلودی خود قرار ندهید

- ساختار نمونه آپلود قابل قبول

```
OS_CAS_81019xxxx.zip
├── parallel
│   ├── main.cpp
│   └── makefile
└── serial
    ├── main.cpp
    └── makefile
```

- برنامه شما باید در سیستم عامل لینوکس و با مترجم g++ با استاندارد C++11 و در زمان معقول برای ورودی‌های آزمون اجرا شود
- دقت کنید که پروژه شما باید دارای Makefile باشد. همچنین در خود مشخص کنید که از استاندارد C++11 استفاده می‌کنید
- نام فایل اجرایی شما که در کنار Makefile ساخته می‌شود باید ImageFilters.out باشد
- نکته‌هایی که در جلسه توجیهی تمرين گفته می‌شود و یا در فرودم‌های مربوطه مطرح می‌شود بخشنی از تمرين هستند؛ بنابراین به آنها توجه داشته باشید
- هدف این تمرين یادگیری شمامست. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد

¹¹ Standard Output Stream