دانشکده مهندسی کامپیوتر

تمرین سری ۳ بینایی کامپیوتر

نام درس

مبانی بینایی کامپیوتر

نام دانشجو

زهرا انوریان

نام استاد درس

دکتر محمدی

پاییز ۱۳۹۹

سوال ۱: در شکل زیر ماتریس بالا، فیلتر و ماتریس پایین، تصویر هستند. برای zero-padding چند ردیف صفر به هر سمت از تصویر اضافه میگردد؟ با در نظر گرفتن zero-padding، تصویر را با فیلتر کانوالو کنید. در حالت BOARDER_REFLECT تصویر را با فیلتر کانوالو کنید و نتیجه را با حالت قبل مقایسه کنید. (۲۰)

۱	۳	۱
۱	۰	۱
۱	۲	۱

فیلتر

۱	۲	۱	۶
۷	۱	۱	۱
۳	۱	۲	۰
۱	۴	۰	۲

تصویر

پاسخ: ابتدا zero padding را به تصویر اعمال می‌کنیم و یک دور به اطراف تصویر پیکسل‌هایی با مقدار صفر اضافه می‌کنیم و برای کانولوشن نیز باید ابتدا فیلتر را ۱۸۰ درجه بچرخانیم و سپس با تصویر کانوالو کنیم.

۰	۰	۰	۰	۰	۰
۰	۱	۲	۱	۶	۰
۰	۷	۱	۱	۱	۰
۰	۳	۱	۲	۰	۰
۰	۱	۴	۰	۲	۰
۰	۰	۰	۰	۰	۰

*

۱	۲	۱
۱	۰	۱
۱	۳	۱

=

۲۴	۱۳	۱۳	۵
۱۵	۲۲	۱۹	۱۶
۲۳	۲۸	۱۱	۱۱
۱۱	۸	۱۱	۲

برای حالت BORDER_REFLECT نیز باید مانند آینه عمل کنیم و خود عدد مرزی را نیز تکرار کنیم. پس داریم:

۱	۱	۲	۱	۶	۶
۱	۱	۲	۱	۶	۶
۷	۷	۱	۱	۱	۱
۳	۳	۱	۲	۰	۰
۱	۱	۴	۰	۲	۲
۱	۱	۴	۰	۲۰	۲

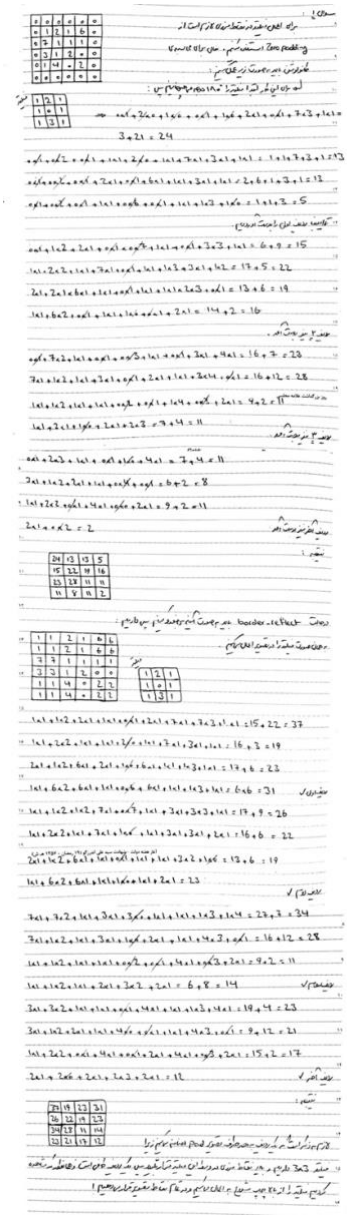
*

۱	۲	۱
۱	۰	۱
۱	۳	۱

=

۳۷	۱۹	۲۳	۳۱
۲۶	۲۲	۱۹	۲۳
۳۴	۲۸	۱۱	۱۴
۲۳	۲۱	۱۷	۱۲

لازم به ذکر است که راه‌حل تمام مراحل در عکس زیر درج شده است.



سوال ۲: به صورت مفهومی توضیح دهید دو کرنل زیر چه پردازشی بر روی تصویر انجام می‌دهند (۱۰)؟

a.

۱	۰	-۱

b.

۱
۲
۱

پاسخ: فیلتر الف خطوط عمودی را برجسته می‌کند زیرا همانطور که می‌دانیم این فیلتر نشان می‌دهد که تصویر سمت راست از چپ تیره‌تر است پس با اعمال این فیلتر، تصویر حاصل دارای خطوط عمودی برجسته‌ای است و در فیلتر ب، فیلتر میانگین وزن‌دار بین هر پیکسل با پیکسل‌های اطرافش می‌گیرد پس تصویر را تار می‌کند اما به دلیل دو برابر بودن پیکسل وسط نسبت به اطراف، این تارشدگی شدت زیادی ندارد.

سوال ۳: فرض کنید تصویر زیر یک تصویر سطح خاکستری است(محدوده رنگ ممکن بین صفر تا ۲۵۵ است .(به صورت مرحله به مرحله الگوریتم CLAHE را بر روی این تصویر با اندازه پنجره 3 × 3 و حالتهای ClipLimit=1 و ClipLimit=2 اعمال کنید(۲۰)

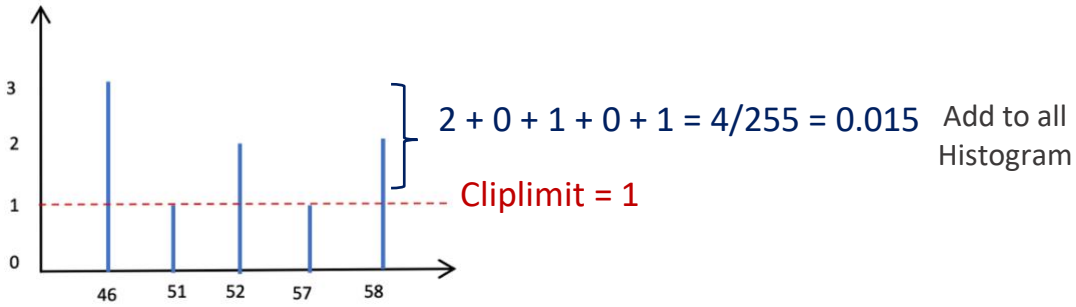
۴۶	۵۱	۵۷	۵۹
۴۶	۵۲	۵۸	۶۰
۴۶	۵۲	۵۸	۶۰

پاسخ: ابتدا باید هیستوگرام را محاسبه کنیم اما به دلیل زیاد بودن رنگ‌ها، همین رنگ‌های استفاده شده را محاسبه می‌کنیم.

	46	51	52	57	58	59	60
n_k	3	1	2	1	2	1	2

و سپس برای اعمال فیلتر بر روی ۹تا پیکسل سمت چپ تصویر به مرکز ۵۲ و به دست آوردن CLAHE باید با توجه به نمودار هیستوگرام بدست آمده و همچنین cliplimit = 1 آن را برش دهیم و مقادیر اضافی را با هم جمع کرده و به هیستوگرام تمام رنگ‌ها اضافه می‌کنیم. پس داریم:

۴۶	۵۱	۵۷
۴۶	۵۲	۵۸
۴۶	۵۲	۵۸



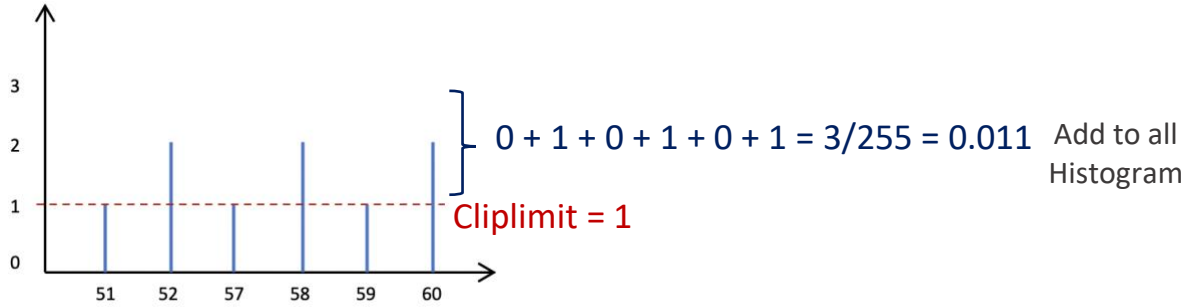
حال باید هیستوگرام را متعادل کنیم. پس داریم:

	0	...	45	46	...	51	52	...	57	58	59	60	...	255
n_k	0.015	...	0.015	1.015	...	1.015	1.015	...	1.015	1.015	0.015	0.015	...	0.015

$\sum n_j$	0.015	…	0.69	1.705	…	2.78	3.795	…	4.87	5.885	5.9	5.915	…	9
$\sum \frac{n_j}{n}$	0.001	…	0.077	0.189	…	0.308	0.421	…	0.541	0.653	0.656	0.657	…	1
$(L-1)\sum \frac{n_j}{n}$	0.255	…	19.55	48.30	…	78.77	107.52	…	137.98	166.74	167.16	167.59	…	255
Round	0	…	20	48	…	79	108	…	138	167	167	168	…	255

همین کار را برای ۹تا پیکسل سمت راست تصویر به مرکز ۵۸ انجام می‌دهیم. پس داریم:

۵۱	۵۷	۵۹
۵۲	۵۸	۶۰
۵۲	۵۸	۶۰



حال باید هیستوگرام را متعادل کنیم. پس داریم:

	0	…	50	51	52	…	57	58	59	60	61	…	255
n_k	0.011	…	0.011	1.011	1.011	…	1.011	1.011	1.011	1.011	0.011	…	0.011
$\sum n_j$	0.011	…	0.561	1.572	2.583	…	3.638	4.649	5.66	6.671	6.682	…	9
$\sum \frac{n_j}{n}$	0.001	…	0.062	0.174	0.287	…	0.404	0.516	0.629	0.741	0.742	…	1
$(L-1)\sum \frac{n_j}{n}$	0.255	…	15.89	44.54	73.18	…	103.07	131.72	160.37	189.01	189.32	…	255
Round	0	…	16	45	73	…	103	132	160	189	189	…	255

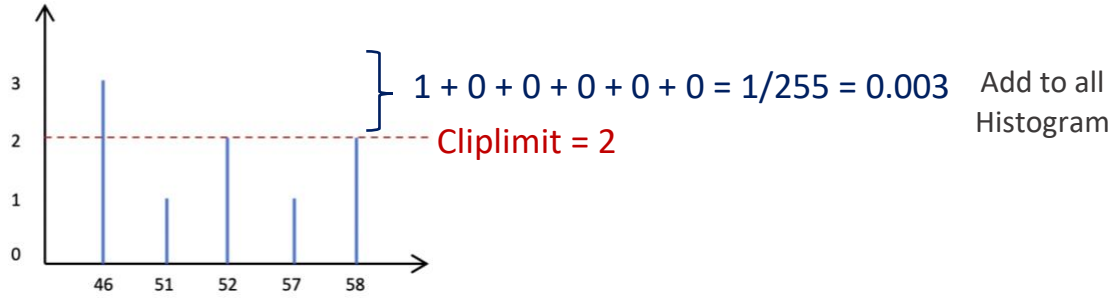
حال برای بدست آوردن نتیجه با محاسبات کمتر (چون باید برای تمام پیکسل‌های تصویر این عملیات CLAHE را انجام دهیم)، می‌توان برای نقاط مرزی باتوجه به t پیکسل مرکزی مقادیرشان را جایگزین کنیم. پس نتیجه‌ی حاصل شده از cliplimit = 1 برابر است با:

۴۸	۷۹	۱۰۳	۱۶۰
۴۸	۱۰۸	۱۳۲	۱۸۹
۴۸	۱۰۸	۱۳۲	۱۸۹

حال می‌خواهیم برای cliplimit = 2 این عملیات را تکرار کنیم. برای ۹تا پیکسل سمت چپ تصویر به مرکز ۵۲ داریم:

۴۶	۵۱	۵۷
۴۶	۵۲	۵۸
۴۶	۵۲	۵۸

ابتدا نمودار هیستوگرام را رسم می‌کنیم و سپس آن را برش می‌دهیم.

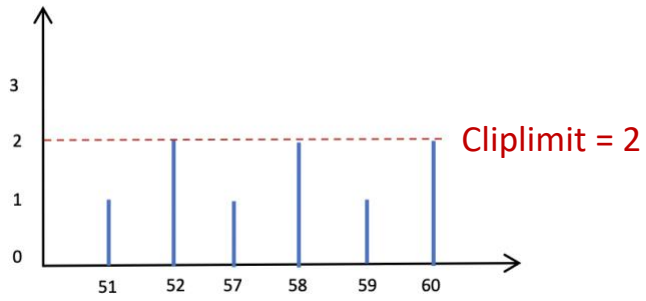


حال باید هیستوگرام را متعادل کنیم. پس داریم:

	0	...	45	46	...	51	52	...	57	58	59	60	...	255
n_k	0.003	...	0.003	2.003	...	1.003	2.003	...	1.003	2.003	0.003	0.003	...	0.003
$\sum n_j$	0.003	...	0.138	2.141	...	3.156	5.159	...	6.174	8.177	8.18	8.183	...	9
$\sum \frac{n_j}{n}$	0.0003	...	0.015	0.237	...	0.350	0.573	...	0.686	0.908	0.9088	0.909	...	1
$(L-1)\sum \frac{n_j}{n}$	0.0765	...	3.91	60.66	...	89.42	146.17	...	174.93	231.68	231.76	231.85	...	255
Round	0	...	4	61	...	89	146	...	175	232	232	232	...	255

همین کار را برای ۹تا پیکسل سمت راست تصویر به مرکز ۵۸ انجام می‌دهیم. پس داریم:

۵۱	۵۷	۵۹
۵۲	۵۸	۶۰
۵۲	۵۸	۶۰



حال باید هیستوگرام را متعادل کنیم. پس داریم:

	0	...	50	51	52	...	57	58	59	60	61	...	255
n_k	0	...	0	1	2	...	1	2	1	2	0	...	0
$\sum n_j$	0	...	0	1	3	...	4	6	7	9	9	...	9
$\sum \frac{n_j}{n}$	0	...	0	0.111	0.34	...	0.45	0.67	0.78	1	1	...	1
$(L-1)\sum \frac{n_j}{n}$	0	...	0	28.34	85	...	113.34	170	198.34	255	255	...	255
Round	0	...	0	28	85	...	113	170	198	255	255	...	255

حال برای بدست آوردن نتیجه با محاسبات کمتر (چون باید برای تمام پیکسل‌های تصویر این عملیات CLAHE را انجام دهیم)، می‌توان برای نقاط مرزی باتوجه به t پیکسل مرکزی مقادیرشان را جایگزین کنیم. پس نتیجه‌ی حاصل شده از cliplimit = 2 برابر است با:

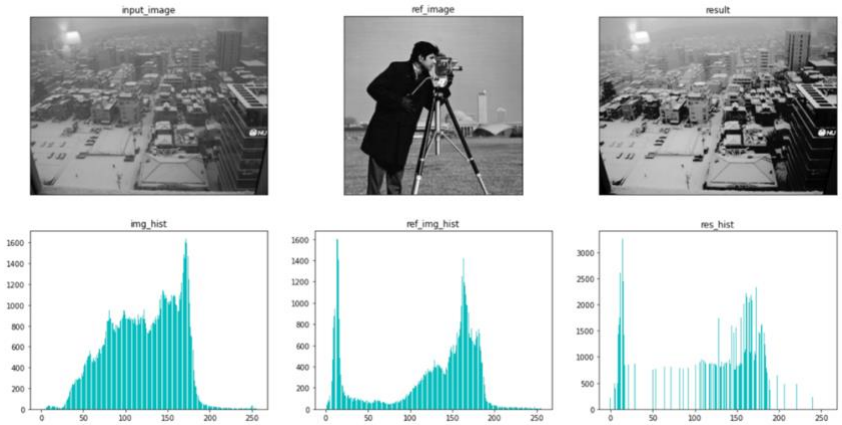
۶۱	۸۹	۱۱۳	۱۹۸
۶۱	۱۴۶	۱۷۰	۲۵۵
۶۱	۱۴۶	۱۷۰	۲۵۵

ادامه سوال۳: برای این سوال ممکن است بخواهید برای ارزیابی پاسخ خود، پیاده‌سازی‌های انجام دهید ولی نتیجه‌های که در پیاده سازی به دست میاورید متفاوت با نتیجه‌های است که حساب خواهید کرد. بررسی کنید که چرا پاسخ شما و پاسخ پیاده‌سازی‌هایی که کردید متفاوت است(۲۰)؟

پاسخ: حال با مقایسه‌ی نتیجه‌ی خود با نتیجه‌ی کد موجود در اینجا متوجه تفاوت شدم که یکی از دلایل آن استفاده از Border_Reflect_101 است که من در حل خود همانطور که پیشتر ذکر شد، برای نقاط مرزی با توجه به t مرکزی مقادیر را جایگزین کردم و padding ای به تصویر ندادم و همچنین در کد opencv برای کم کردن محاسبات و در نتیجه بیشتر شدن سرعت از Interpolation نیز استفاده کرده است که باتوجه به جستجویی که کردم به این صورت است که تصویر را به چند قسمت با ابعاد مشخص (به طور مثال 5x5) تقسیم می‌کند و بر پیکسل مرکزی CLAHE را اعمال می‌کند و برای بدست آوردن بقیه پیکسل‌های اطراف از Linear Interpolation یا bilinear Interpolation استفاده می‌کند و مقادیر بدست آمده را جایگزین می‌کند. Linear Interpolation برای پیکسل‌هایی که دو همسایگی دارند و biliner Interpolation برای پیکسل‌هایی است که چهار همسایگی دارند استفاده می‌شود.

سوال ۴: گاهی برای کارهای پردازش تصویر لازم است هیستوگرام یک تصویر را شبیه به هیستوگرام یک تصویر دیگر یا هیستوگرام از پیش تعیین شده بکنیم برای این کار از تطبیق هیستوگرام استفاده می‌کنیم که در آن از تابع متعادل‌سازی هیستوگرام تصویر ورودی استفاده می‌کنیم و معکوس تابع متعادل سازی هیستوگرام تصویر مرجع را بر روی ان اعمال می‌کنیم. برای تصویر Q4.jpg و تصویر مرجع Q4_ref.jpg الگوریتم تطبیق هیستوگرام را بدون توابع پیش ساخته پیاده‌سازی کنید. برای این کار تابع histogram_matching را کامل کنید. در این تابع ورودی تصویر اولیه و تصویر مرجع هستند و خروجی تصویر حاصل از تطبیق هیستوگرام تصویر اولیه بر تصویر مرجع است. تصاویر ورودی را به صورت تک کاناله بخوانید. در تصویر پایین یک نمونه از چیزی که مورد انتظار است برای ورودی‌های دیگری غیر از ورودی‌های این سوال، امده است. شاید در خود تصویر تغییرات اعمال شده محسوس نباشد ولی در هیستوگرام‌ها مشخص است که چه کاری صورت گرفته است(۳۰).

پاسخ: برای histogram matching باید cdf تصویر ورودی و تصویر مرجع را بدست آورد و سپس به ازای cdfهای یکسان matching را انجام می‌دهیم. لازم به ذکر است که ممکن است برای مقادیری cdf یکسان موجود نباشد که در این صورت اولین cdf مرجع بزرگتر از cdf تصویر را جایگذاری می‌کنیم.



سوال ۵: در این تابع بدون استفاده از توابع آماده در OpenCV کرنل مد نظر را ساخته با استفاده از تابع filter2D از کتابخانه OpenCV آن را با تصویر کانوالو کنید. بار دیگر تصویر را با استفاده از تابع GaussianBlur در کتابخانه OpenCV فیلتر کرده و نتایج را از نظر کیفیت و سرعت با هم مقایسه کنید. خروجی مورد انتظار برای این سوال این است که دو خروجی تقریبا مشابه هم شوند .برای این سوال تصویر img5.jpg را به صورت تک کاناله خوانده و تغییرات بالا را روی ان اعمال کنید .برای نحوه استفاده از توابع filter2D و GaussianBlur از کتابخانه OpenCV به این لینک مراجعه کنید(۲۰).

پاسخ: با توجه به فرمول گوسی زیر کرنل را باید بدست بیاوریم. حال مقادیر x و y داخل فرمول برابر با مختصات فیلتر ما می‌باشد که به طور مثال برای فیلتر گوسی با سایز 3x3 فیلتر دارای نقاط 1- تا 1 در هر دو جهت x و y است که من برای جنرال کردن تابع که برای سایزهای مختلف نیز کرنل مناسبی را تولید کند این حدود را به صورت تقسیم سایز بر دو نوشتم و از آنجایی که سایز کرنل باید عددی فرد باشد آن را رند به پایین کردم و همچنین برای تبدیل درست مختصات کرنل با آرایه مقادیر x و y را با رند بالای آن جمع کردم.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

```
time for applying your guassian filter: 0.009897 s
time for applying OpenCV guassian filter: 0.005666 s
```

با توجه به مقایسه‌ی نتیجه‌ی این دو تابع، در کیفیت آن‌ها با هم تفاوت زیادی مشاهده نمی‌شود اما همانطور که مشاهده می‌کنید سرعت تابع opencv از تابع پیاده‌سازی شده کمی بیشتر است.