



تمرین عملی سری دوم پردازش تصاویر دیجیتال

نام و نام خانوادگی: امیرمهدی جعفری فشارکی

شماره دانشجویی: ۹۸۱۰۹۶۴۵

۱۴۰۱ فروردین

سوال اول

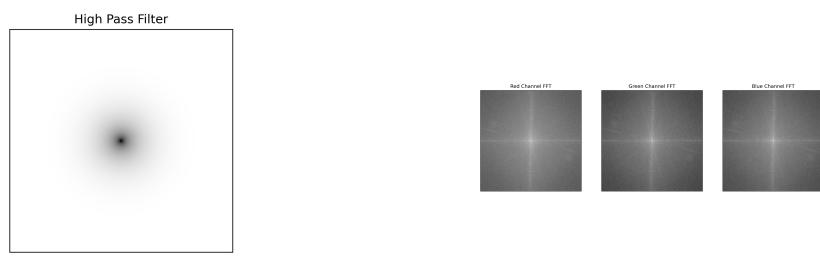
روش اول

فرض کنید تصویر اصلی را قبل از blur شدن با $f_i(x, y)$ نمایش دهیم. در این صورت برای blur کردن تصویر از یک فیلتر پایین گذر که آن را با $l(x, y)$ نمایش می‌دهیم استفاده شده است. در این صورت تصویر blur شده فعلی ای که ما داریم برابر است با $f(x, y) = f_i(x, y) * l(x, y)$. در نتیجه تصویر فعلی ما در حوزه فرکانس برابر است با $F(u, v) = F_i(u, v)L(u, v)$. از آنجایی که $L(u, v)$ یک فیلتر پایین گذر بوده، این به این معنی است که در حال حاضر بخش‌های فرکانس بالای عکس اصلی تضعیف و بخش‌های فرکانس پایین (که معادل تغییرات کم در تصویر می‌باشد) تقویت شده است. به عبارت دیگر $f_i(x, y) - f(x, y)$ دارای بخش‌های فرکانس بالای تصویر می‌باشد. به همین دلیل برای بازسازی و تیزتر کردن تصویر ما باید $f(x, y)$ را با بخش‌های فرکانس بالای تصویر اصلی که تضعیف شده‌اند جمع کنیم. اما از آنجا که به این بخش‌ها از تصویر اصلی $f_i(x, y) - f(x, y)$ دست‌پیدا کنیم، می‌توانیم با اعمال یک فیلتر بالاگذر بر روی تصویر $f(x, y)$ به تقریبی از بخش‌های فرکانس بالا یا همان $f_i(x, y) - f(x, y)$ دست‌پیدا کنیم. در نتیجه داریم:

$$\hat{f}_{hp} = \mathcal{F}^{-1}\{H_{HP}(u, v).F(u, v)\} \quad (1)$$

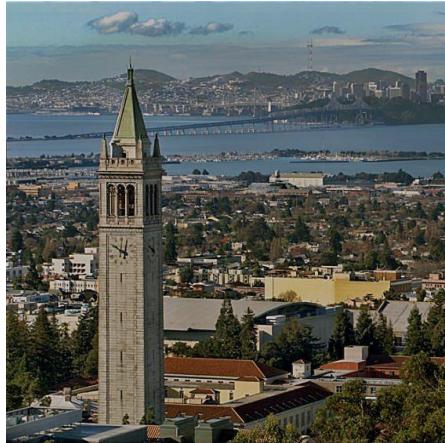
$$\begin{aligned} \hat{f}_i &= f + k\hat{f}_{hp} = \mathcal{F}^{-1}\{F(u, v) + H_{HP}(u, v)F(u, v)\} \\ \implies \hat{f}_i &= \mathcal{F}^{-1}\{(1 + kH_{HP}(u, v))F(u, v)\} \end{aligned} \quad (2)$$

این روش پیاده‌سازی شده و نتایج مربوط به آن به صورت فایل‌های خواسته شده ذخیره شده است. تصاویر خواسته شده مطابق زیر می‌باشند. همچنین برای دست یافتن به این نتیجه از $k = 1$ استفاده شده است.

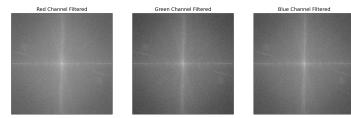


(آ) اندازه تبدیل فوریه تصویر (ب) اندازه تبدیل فوریه فیلتر بالاگذر

شکل ۱: اندازه تبدیل فوریه تصویر و فیلتر بالاگذر



(ب) تصویر نهایی

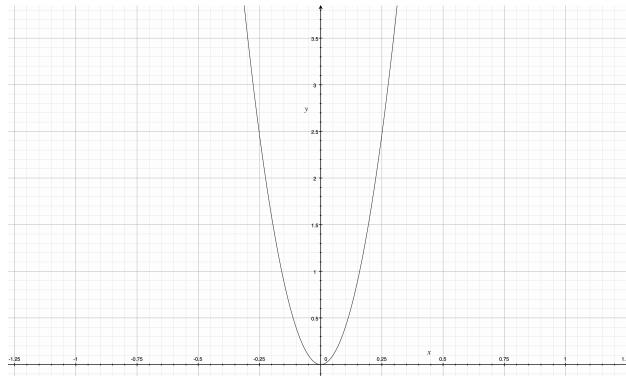


(ا) اندازه تبدیل فوریه تصویر نهایی

شکل ۲: تصویر نهایی و تبدیل فوریه آن

روش دوم

در صورتی که تعریف کنیم $r = \sqrt{u^2 + v^2}$ ، تابع $r = \sqrt{u^2 + v^2}$ به شکل $g(r) = 4\pi^2 r^2 = 4\pi^2(u^2 + v^2)$ در می‌آید که نمودار آن مطابق با شکل زیر می‌باشد. همان‌طور که در شکل زیر مشاهده می‌شود، با افزایش فاصله از مبدأ، مقدار این تابع افزایش می‌یابد و به همین دلیل در عمل این رابطه خود نوعی فیلتر

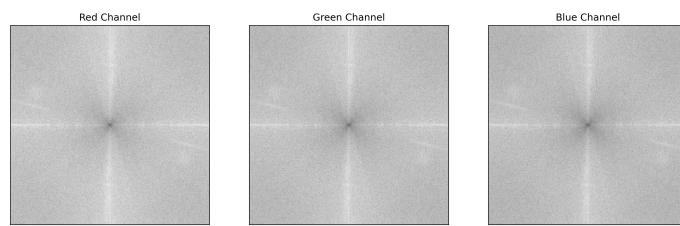


شکل ۳: نمودار تابع $g(r)$

بالاگذر می‌باشد و در نتیجه با جایگذاری این فیلتر در همان رابطه (۲) داریم:

$$\begin{aligned}
 \hat{f}_i &= \mathcal{F}^{-1}\{(1 + kH_{HP}(u, v))F(u, v)\} \\
 &= \mathcal{F}^{-1}\{(1 + k(4\pi^2(u^2 + v^2)))F(u, v)\} \\
 &= f + k\mathcal{F}^{-1}\{4\pi^2(u^2 + v^2)F(u, v)\}
 \end{aligned} \tag{3}$$

با اعمال فیلتر گفته شده و به ازای $k = 0.15$ خروجی های خواسته شده مطابق شکل های زیر خواهند شد.



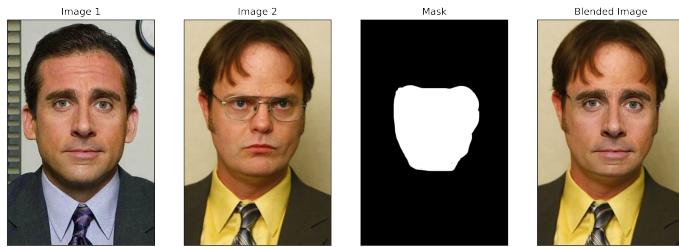
شکل ۴: اندازه تبدیل فوریه unsharp mask



شکل ۵: تصویر نهایی و تبدیل فوریه آن

سوال دوم

برای پیاده‌سازی این سوال از ۴ تابع نوشته شده است. تابع اول یعنی $gaussian_stack(im,i)$ یک تصویر و مرحله مورد نظر در پشته (i) را به عنوان ورودی گرفته و در خروجی پشته گوسی تصویر در مرحله n ام را باز می‌گرداند. به همین طریق $laplacian_stack(im,i)$ پشته لaplاسین مرحله n ام را باز می‌گرداند. نکته قابل توجه این است که $l_{slastlevel} = G_{slastlevel} - G_{slastlevel+1}$ که اندیس last level + 1 وجود ندارد و به عبارت دیگر علاوه بر باز می‌باشد و به همین دلیل $l_{slastlevel} = G_{slastlevel}$ می‌باشد. در حقیقت پشته لaplاسین مرحله آخر، کلیات تصویر (عوامل فرکانس پایین) را در بر دارد و به همین دلیل کمکی که به ترکیب عکس‌ها با هم می‌کند این است که باعث می‌شود کلیات تصویر ترکیب شده میانگینی وزن دار از کلیات تصویر هر دو تصویر باشد که خود به خود باعث smooth تر شدن تصویر ترکیب شده می‌شود. همچنین تابع $blended_stack$ پشته لaplاسین ترکیب عکس‌ها را به وجود آورده و تابع $blend_image$ با استفاده از این تابع، پشته‌های لaplاسین را با هم جمع کرده تا به تصویر نهایی برسد. در نهایت خروجی مطابق شکل زیر می‌باشد.



شکل ۶: تصاویر اولیه، ماسک استفاده شده و تصویر ترکیب شده

سوال سوم

الگوریتم SSR

در حقیقت کاری که این الگوریتم انجام می‌دهد این است که اختلاف میان شدت هر پیکسل و میانگین وزن دار شدت پیکسل‌های مجاور را به عنوان خروجی می‌دهد. رابطه استفاده شده برای هر کanal رنگی به شکل زیر می‌باشد.

$$R_i(x, y) = \log(I_i(x, y)) - \log(I_i(x, y) * F(x, y)) \quad (4)$$

همان‌طور که می‌دانیم، فرضی وجود دارد بر مبنای اینکه شدت هر تصویر تشکیل شده از ضرب شدت نور، $S(x, y)$ ، در میزان بازتاب جسم، $r(x, y)$ ، می‌باشد. یعنی داریم:

$$I_i(x, y) = S_i(x, y)r_i(x, y) \quad (5)$$

حال با استفاده از این موضوع و معادله (4) را بازنویسی می‌کنیم.

$$R_i(x, y) \simeq \log \frac{S_i(x, y)r_i(x, y)}{\bar{S}_i\bar{r}_i}$$

همچنین از آنجایی که نورپردازی محیط تابعی با تغییرات کم می‌باشد می‌توان فرض کرد که میانگین آن به تقریب خوبی برابر با مقدار آن در هر نقطه می‌باشد یا به عبارت دیگر $S_i(x, y) \simeq \bar{S}_i$ و در نتیجه به معادله زیر دست می‌یابیم که نشان دهنده این است که خروجی تابع SSR اثرات سایه را در تصویر تضعیف می‌کند و به عبارت دیگر خروجی وابستگی به شدت نور محیط ندارد.

$$R_i(x, y) \simeq \log \frac{r_i(x, y)}{\bar{r}_i} \quad (6)$$

حال برای پیاده سازی این تابع، برای تابع $F(x, y)$ که در حقیقت در میانگین گیری نقش دارد، گرینه های متعددی موجود است که تابع استفاده شده در مقاله یک تابع گوسی است که با انجام آزمایشات متعدد این نتیجه حاصل شده است که با قرار دادن $\sigma = 80$ در تابع گوسی، خروجی بهتری به نسبت بقیه مقادیر به دست می‌آید. نهایت خروجی به دست آمده برای تصویر داده شده مطابق شکل ۷-آ می‌باشد.

الگوریتم MSR

الگوریتم MSR در حقیقت از میانگین وزن دار الگوریتم SSR با $F(x, y)$ متفاوت به دست می‌آید که به گفته مقاله، با آزمایشات انجام شده، استفاده از ۳ SSR با توابع گوسی که در آنها $\sigma_1 = 15$ ، $\sigma_2 = 80$ و $\sigma_3 = 250$ می‌باشند و با وزن‌های برابر، به نتیجه قابل قبولی ختم می‌شود. شکل ۷-ب خروجی این تابع بر روی تصویر داده شده را نمایش می‌دهد.



(ب) خروجی تابع MSR



(ل) خروجی تابع SSR

شکل ۷: خروجی توابع SSR و MSR

الگوریتم MSRCR

این الگوریتم در حقیقت همان الگوریتم MSR با اصلاح رنگ می‌باشد که برای اصلاح رنگ پس از به دست آوردن مقدار MSR هر کanal، این مقدار در یک تابع اصلاح کننده مطابق معادله (۷) ضرب می‌شود.

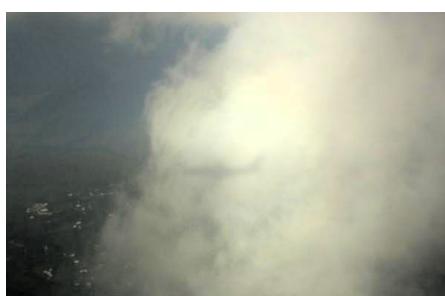
$$C_i(x, y) = f(I'_i(x, y)) \quad (7)$$

در این مقاله این تابع به شکل زیر در نظر گرفته شده است و خروجی این الگوریتم برای تصویر داده شده مطابق شکل ۸-آ می‌باشد.

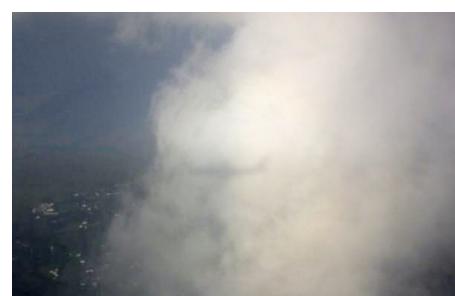
$$C_i(x, y) = \beta \log[\alpha I'_i(x, y)] = 46[\log(125I_i(x, y)) - \log(\sum_{j=1}^n I_j(x, y))] \quad (8)$$

الگوریتم MSRCP

خروجی این الگوریتم برای تصویر داده شده مطابق شکل ۸-ب می‌باشد.



(ب) خروجی تابع MSRCP



(ل) خروجی تابع MSRCR

شکل ۸: خروجی توابع MSRCP و MSRCR

سوال چهارم

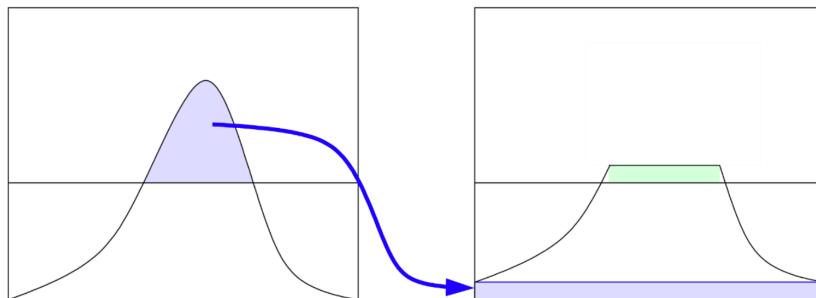
روش HE: در این روش در حقیقت هدف این است که با استفاده از یک نگاشت به شکل $s = T(r)$ که در آن r شدت روشنایی در تصویر اولیه و s شدت روشنایی در تصویر جدید می‌باشد، به توزیعی یکنواخت برای شدت روشنایی تصویر دست پیدا کنیم. در حالت پیوسته ثابت می‌شود در صورتی که نگاشت ما، CDF میزان روشنایی در هر لحظه باشد، به هدف مورد نظر دست پیدا می‌کنیم. اما در حالت گسسته دقیقاً این اتفاق رخ نمی‌دهد و به جای CDF از نگاشت زیر استفاده می‌شود

$$S_k = (L - 1) \sum_{j=0}^k \frac{n_k}{MN} \quad (9)$$

که در آن M و N ابعاد تصویر، L تعداد سطوح روشنایی و n_k تعداد پیکسل‌های با سطح روشنایی k می‌باشند و در نهایت مقدار به دست آمده گرد می‌شود.

روش AHE: در این روش همان الگوریتم HE اجرا می‌شود با این تفاوت که به جای یک نگاشت واحد برای کل تصویر اعمال شود، برای هر پیکسل از تصویر هیستوگرام مربوط به پیکسل‌های محلی که در یک پنجه با ابعاد از پیش تعیین شده حول پیکسل مورد نظر قرار دارند، محاسبه شده و سپس از روی این هیستوگرام‌های که برای هر پیکسل متفاوت می‌باشند، نگاشتی که پیشتر ذکر شد به دست آمده و بر روی آن پیکسل اعمال می‌شود. البته لازم به ذکر است که انجام این کار برای یک تصویر و برای هر پیکسل به صورت جداگانه از لحاظ محاسباتی بهینه نمی‌باشد و به همین دلیل به جای این کار، در ابتدا تصویر را به چند بخش مساوی تقسیم کرده و این نگاشت‌ها را برای پیکسل مرکزی هر کدام از این بخش‌ها هساب می‌کنیم و برای بقیه پیکسل‌ها با استفاده از درون‌یابی نگاشت‌های به دست آمده برای این پیکسل‌های مرکزی هر بخش، نگاشت مربوط به آن پیکسل‌ها را محاسبه می‌کنیم.

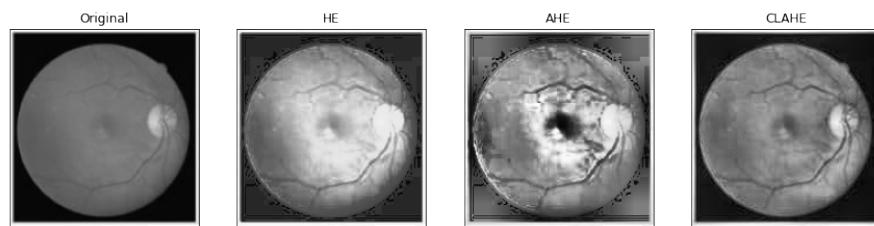
روش CLAHE این روش در حقیقت مشابه AHE عمل کرده و تنها تفاوت آن در محاسبه نگاشت از روی هیستوگرام است به این صورت که پیش از محاسبه نگاشت، مقادیری از هیستوگرام که از یک مقدار مشخص بیشتر می‌باشند حذف شده و به صورت یکنواخت به تمام هیستوگرام اضافه می‌شوند. سپس نگاشت مورد نظر از روی هیستوگرام جدید به دست آمده محاسبه شده و بقیه مراحل مانند AHE طی می‌شود. شکل ۹ نحوه قطع کردن هیستوگرام را نمایش می‌دهد.



شکل ۹: قطع کردن هیستوگرام در روش CLAHE

برای تصاویر داده شده، می‌توان حدس زد که تصویر اول یا همان تصویر سمت چپ، با استفاده از روش CLAHE به دست آمده است چرا که بر اساس این موضوع که روشنایی بخشی از تصویر زیاد شده است و در عین حال بخش‌های تاریک، تا حدودی تاریک مانده است و علاوه بر این، نویز هم در تصویر به آن صورت تقویت نشده است، می‌توان به این نتیجه رسید که الگوریتم استفاده شده CLAHE می‌باشد. همچنین برای تصویر دوم یا همان تصویر سمت راست، می‌توان حدس زد که از روش AHE به دست آمده است چرا که قسمت‌هایی از تصویر که در ابتداء روشنایی کمی داشته است (ناحیه دور تا دور تصویر) بسیار روشن شده است و در عین حال نویز به حد زیادی تقویت شده است که می‌توان نتیجه گرفت از الگوریتم AHE استفاده شده است.

خروجی روشن‌های خواسته شده بر روی تصویر q4.png مطابق شکل زیر می‌باشد.

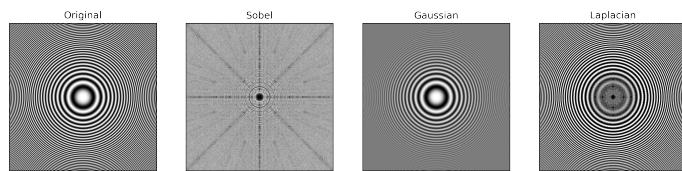


شکل ۱۰ : خروجی‌های کد

سوال پنجم

بخش اول

خروجی فیلترهای خواسته شده مطابق شکل ۱۱ می‌باشد.



شکل ۱۱ : مقایسه خروجی فیلترها با تصویر اصلی

همان‌طور که مشاهده می‌شود، در خروجی فیلتر گوسی، بخش‌های دور تصویر که متشکل از نوارهای با فرکانس بالاتر می‌باشند، تا حدی کمرنگ و تصعیف شده‌اند.

در خروجی فیلتر لابلائسین، لبه‌های تصویر تیز تر و تصویر Sharp تر شده است همچنین پهنه‌ای هر کدام از نوارهای دایروی کمی کمتر شده است نکته جالب این است که در وسط تصویر (که در تصویر اصلی یک دایره سفید است) به جای دایره سفید، دایره کوچکتری با رنگ سیاه به وجود آمده است. دلیل این رفتار فیلتر لابلائسین این است که فیلتر لابلائسین در مشتق دوم را را نمایش می‌دهد و به دلیل اینکه در وسط تصویر و در بازه مشخصی میزان روشنایی تقریباً ثابت است، مشتقات اول و دوم هردو صفر بوده و به همین دلیل دایره تیزه رنگ در وسط صفحه تشکیل شده است.

فیلتر سوبل نیز نقش تشخیص لبه در تصویر را انجام می‌دهد و همان‌طور که مشاهده می‌شود، در نقاط مرزی لبه‌های تصویر، خروجی فیلتر سوبل روشنایی بیشتر و در نقاط وسط هر کدام از نوارهای تصویر اصلی، خروجی فیلتر سوبل تیزه تر می‌باشد که کاملاً منطقی می‌باشد.

بخش دوم

برای اثبات گوسی بودن تبدیل فوریه فیلتر گوسی از آنجا که فیلتر گوسی جدایی‌پذیر است، کافیست برای یک فیلتر گوسی یک بعدی این موضوع را اثبات کنیم و در این صورت برای فیلتر گوسی دو بعدی نیز اثبات می‌شود:

$$\begin{aligned}
 F(u) &= \int \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} e^{-j2\pi ux} dx \\
 \frac{dF(u)}{du} &= -\frac{j\sqrt{2\pi}}{\sigma} \int e^{-j2\pi ux} x e^{-\frac{x^2}{2\sigma^2}} dx \\
 &= \frac{j\sqrt{2\pi}}{\sigma} \int e^{-j2\pi ux} d(\sigma^2 e^{-\frac{x^2}{2\sigma^2}}) \\
 &= j\sqrt{2\pi}\sigma \left[e^{-j2\pi ux - \frac{x^2}{2\sigma^2}} \Big|_{-\infty}^{\infty} + j2\pi u \int e^{-\frac{x^2}{2\sigma^2}} e^{-j2\pi ux} dx \right] \\
 &= -4\pi^2\sigma u F(u) \\
 \implies \frac{dF(u)}{F(u)} &= -2\pi^2\sigma^2 d(u^2) \implies \ln\left(\frac{F(u)}{F(0)}\right) = -2\pi^2\sigma^2 u^2 \\
 \implies F(u) &= F(0)e^{-2\pi^2\sigma^2 u^2} \\
 F(0) &= \int \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} dx = \sqrt{2\pi}\sigma \\
 \implies F(u) &= \sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2 u^2} \tag{11}
 \end{aligned}$$

و در نتیجه اثبات، بالا برای حالت دو بعدی خواهیم داشت:

$$F(u, v) = 2\pi\sigma^2 e^{-2\pi^2\sigma^2(u^2+v^2)} \tag{12}$$

در ادامه، نتایج خواسته شده در فایل‌های ذکر شده ذخیره شده‌اند. همچنین تصویر به دست آمده ناشی از اعمال فیلتر گوسی تفاوتی با تصویر ناشی از کانوالوکردن فیلتر در حوزه زمان که در قسمت قبل به دست آورده‌یم ندارد. در رابطه با متمم فیلتر گوسی می‌توان مشاهده کرد که تصویر خروجی در نقاط وسط که فرکانس بالاتری دارد تیره‌تر شده است و همچنین نوارهای دور صفحه واضح‌تر و تیز‌تر شده‌اند که خود تاییدی بر این است که این فیلتر متمم گوسی یک فیلتر بالاگذر است.

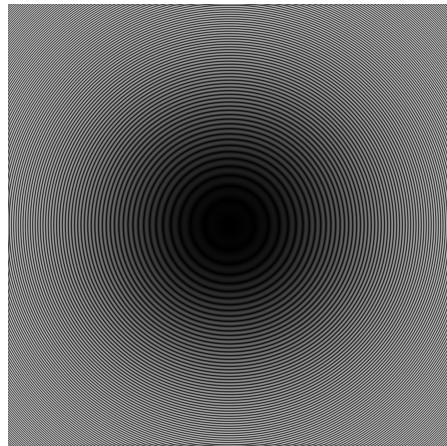
در مقابل، با اعمال فیلتر مربعی مشاهده می‌کنیم که باز هم بخش‌ها فرکانس بالا حذف شده اما نکته قابل توجه این است که شکل کلی تصویر هم به شکل مربعی در می‌آید. و باز هم با اعمالاً متمم مشاهده می‌شود که بخش‌های وسط تصویر که دارای فرکانس پایین هستند کاملاً از بین رفته اما به دلیل مربعی بودن خود فیلتر بخش‌های فرکانس بالا (دور تا دور تصویر) نیز دچار تغییراتی می‌شوند.

دلیل به وجود آمدن مربع‌های کوچک روی تصویر این می‌باشد که زمانی که ما در حوزه فرکانس یک فیلتر مربعی اعمال می‌کنیم، این فیلتر معادل ضرب دو Sinc یکی بر حسب x و دیگری بر حسب y می‌شود. به همین دلیل، از آن جا که در حوزه زمان این فیلتر کانوالو می‌شود، انگار که در هر نقطه، در برخی نقاط برای‌یند مقادیر ضرب شده در دره و قله‌های سینک مقدار زیاد و در برخی مقداری کم می‌شود و برای مثال در بخش وسط تصویر که کاملاً سفید است، زمانی که از این فیلتر رد می‌شود، با جابجا کردن پنجره قله‌ها و دره‌ها به صورت متوالی ضرایب زیاد و کم می‌گیرند که به همین دلیل باعث

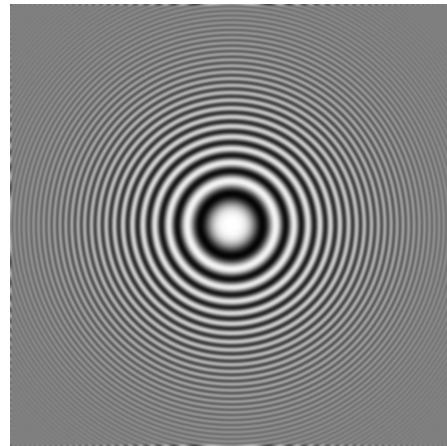
به وجود آمدن یک pattern متناوت سیاه و سفید روی تصویر در دو راستا می‌شود که در نهایت مانند مربع‌هایی کوچک در تصویر ظاهر می‌شوند.

بخش سوم

با بزرگنمایی تصویر بخش‌های فرکانس پایین تصویر بیشتر دیده می‌شوند و به همین دلیل کامل‌تر دیده می‌شوند و با کوچکتر کردن عکس، دایره‌های کوچکی در گوش‌های تصویر نیز دیده می‌شوند.

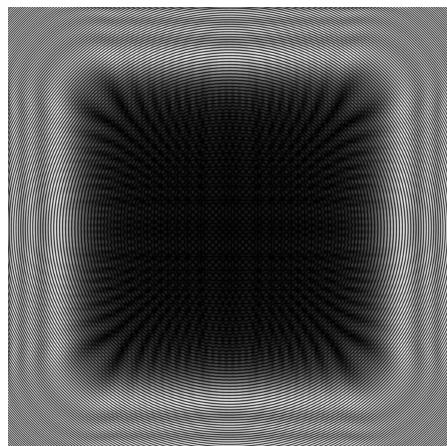


(ب) خروجی متمم گوسی

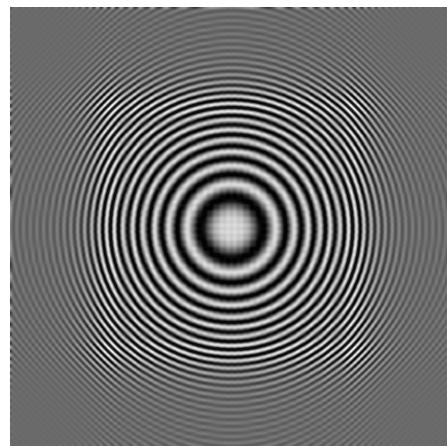


(آ) خروجی فیلتر گوسی

شکل ۱۲ : خروجی های فیلتر گوسی و متمم آن



(ب) خروجی متمم مربعی



(آ) خروجی فیلتر مربعی

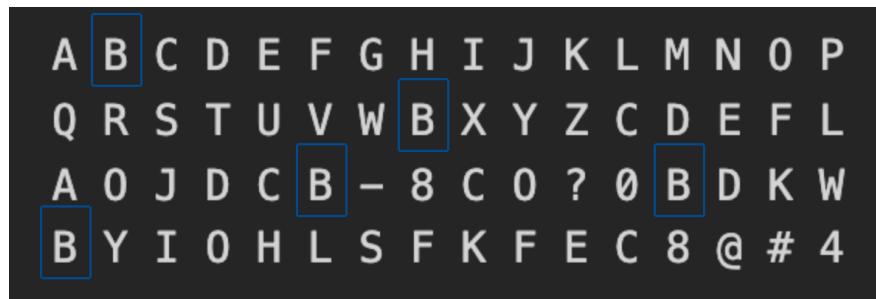
شکل ۱۳ : خروجی های فیلتر مربعی و متمم آن

سوال ششم

دو روش که در کد پیاده‌سازی شده از آن‌ها استفاده شده، روش Pyramid Processing و Normalized Template Correlation می‌باشد که در روش اول، ابتدا توسط تابع `corr_array_calc`، همبستگی نرمالیزه شده عکس اصلی و `template` برای تصویر اصلی در هر پیکسل و با اعمال یک پنجره با سایز `template` از رابطه زیر محاسبه می‌شود.

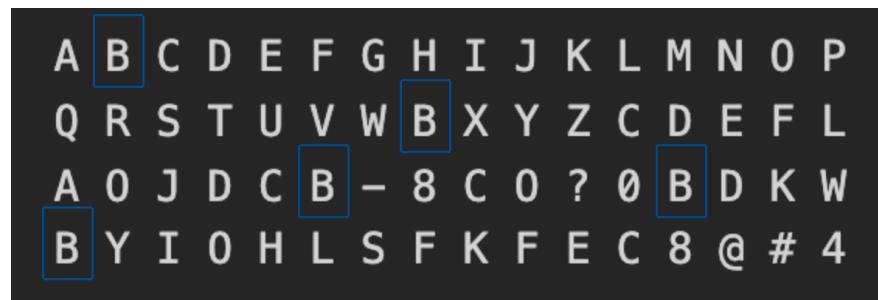
$$NCC(Image1, Image2) = \frac{1}{N\sigma_1\sigma_2} \sum_{x,y} (Image1(x,y) - \overline{Image1}) \times (Image2(x,y) - \overline{Image2}) \quad (13)$$

در ادامه، با قرار دادن یک آستانه که در کد من برابر با 95.0 می‌باشد، پیکسل‌هایی که دارای NCC بیشتر از این مقدار می‌باشند انتخاب شده و سپس توسط تابع `local_maxima_detector` از بین این پیکسل‌ها، آنهایی که ماسکیم محلی هستند انتخاب شده و در نهایت یک مستطیل به بعد template دور هر کدام از این پیکسل‌ها کشیده می‌شود. خروجی این روش مطابق شکل زیر می‌شود.



شکل ۱۴ : خروجی روش Template Correlation

در روش Pyramid Processing در حقیقت ابتدا تصاویر (هم تمپلیت و هم تصویر اصلی) کوچک می‌شوند. در هر مرحله هرم، طول و عرض تصویر هر کدام نصف می‌شوند. در روش پیاده‌سازی شده در کد، تصاویر تا ۲ مرحله کوچک شده‌اند. در حقیقت تابع `pyramid_processing` با گرفتن دو تصویر به عنوان ورودی و تعداد مراحل کوچکنمایی، ابتدا با استفاده از تابع `image_pyramid` یک آرایه از لایه‌های مختلف هرم برای هر دو عکس درست می‌کند. سپس از کوچکترین عکس شروع کرده و مطابق روش قبل برای آن ماتریس NCC را تشکیل داده و سپس با گذاشتن یک آستانه (که در اینجا 8.0 می‌باشد) مختصات پیکسل‌هایی که از آن NCC آن‌ها از آن آستانه بیشتر باشد را به دست آورده. در ادامه این مختصات را در دو ضرب کرده و مختصاتی جدید متناظر با نصوبی که یک لایه کمتر کوچکنمایی شده است به دست می‌آید. سپس بار دیگر NCC را محاسبه کرده اما اینبار فقط برای مختصات به دست آمده و 8 همسایه هر کدام از این مختصات این محاسبات را انجام می‌دهد و به همین طریق با ادامه این روند و آستانه گذاری و محاسبه مختصات جدید، این کار را تا جایی که به لایه 0 هرم برسد ادامه می‌دهد. اینبار به جای آستانه 8.0 ، مانند بخش قبل آستانه 95.0 را اعمال می‌کند و در نهایت دور نقاط به دست آمده خط می‌کشد. خروجی این روش نیز مطابق شکل زیر می‌شود.



شکل ۱۵ : خروجی روش Pyramid Processing

سوال هفتم

تصاویر استفاده شده همان تصاویر سوال دو می‌باشند(که هم ابعاد آن و هم چشم‌های آن‌ها روی هم قرار دارد). در ابتدا این تصاویر از RGB به Grayscale تبدیل شده‌اند. این تصاویر مطابق شکل زیر می‌باشند:



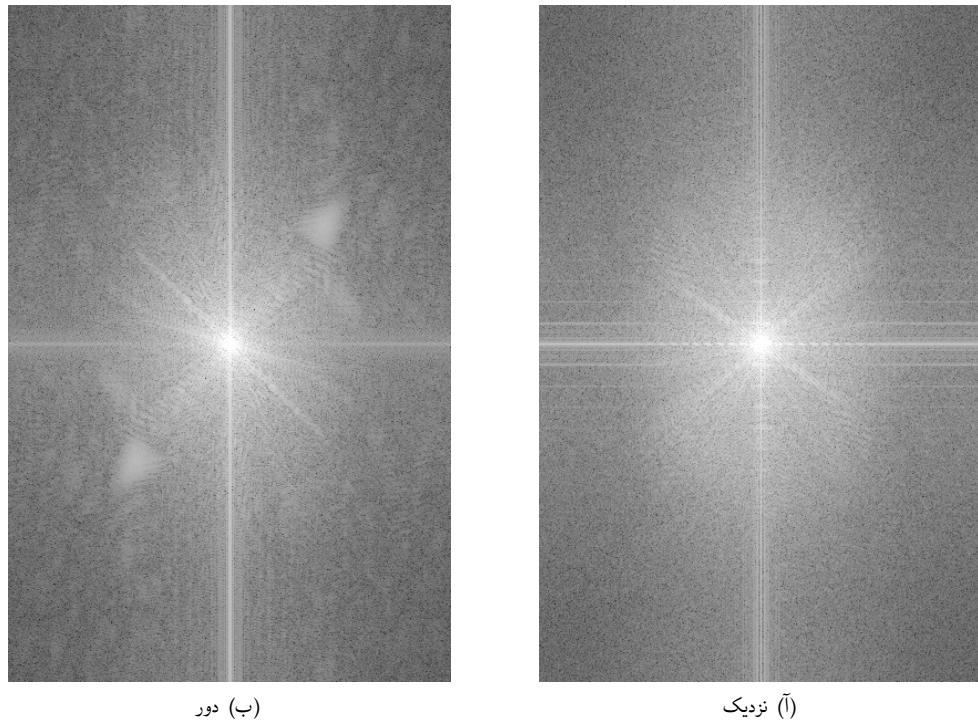
(ب) دور



(آ) نزدیک

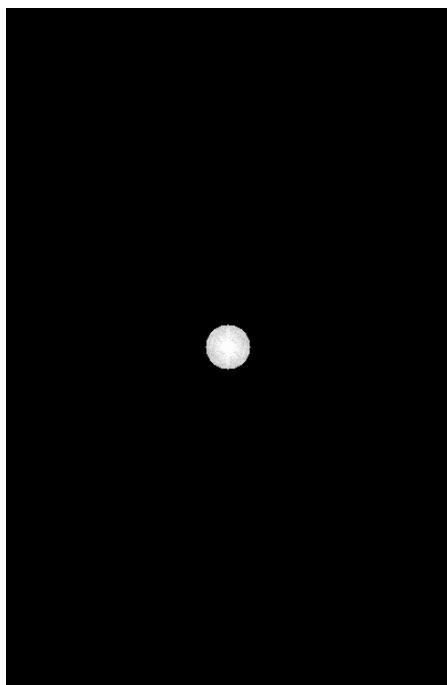
شکل ۱۶ : تصاویر اولیه

در ادامه، تبدیل فوریه هر کدام از این تصاویر محاسبه شده و مطابق شکل زیر می‌باشند.

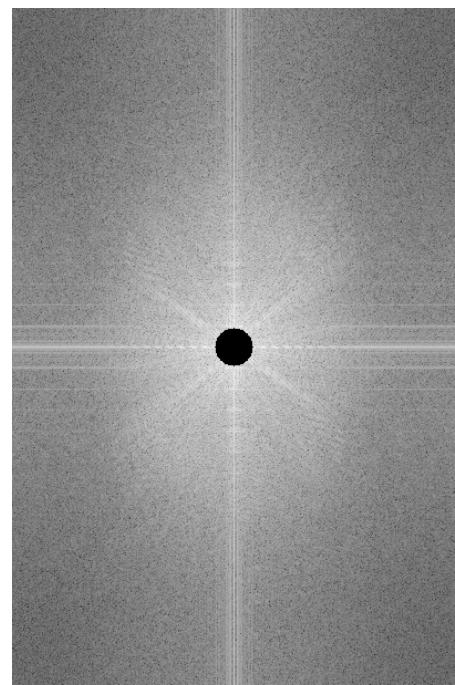


شکل ۱۷ : تبدیل فوریه تصاویر اولیه

پس از محاسبه تبدیل فوریه، به سراغ تولید فیلترهای مورد نظرمان می‌رویم. برای اینکار تابعی با نام gaussianFilter نوشته شده است که ابعاد تصویر و سیگما فیلتر مورد نظر و بالا یا پایین‌گذر بودن فیلتر را به عنوان ورودی گرفته و در خروجی فیلتر گوسی با مشخصات داده شده در حوزه فرکانس تولید می‌کند. با استفاده از این دو تابع دو فیلتر یکی بالاگذر با سیگما برابر با ۱۵ و دیگری پایین‌گذر با سیگما برابر با ۱۷ تولید کرده ایم. سپس یک با استفاده از تابع cutoffFilter دو فیلتر هم ایجاد می‌کنیم و حال این فیلتر و فیلترهای گوسی را یکی بر روی تصویر دور و دیگری بر روی تصویر نزدیک اعمال می‌کنیم و خروجی فیلتر شده دو تصویر در حوزه فرکانس مطابق شکل صفحه بعد می‌شوند.



(ب) دور



(آ) نزدیک

شکل ۱۸: تبدیل فوریه تصاویر فیلتر شده

در نهایت با گرفتن تبدیل فوریه معکوس و جمع کردن دو خروجی به دست آمده به تصویر نهایی در صفحه بعد می‌رسیم که همان تصویر هیبرید مورد نظر می‌باشد.



شكل ١٩: تصویر هیبرید نهایی
١٦