

تمرین سری **چهارم**

درس تصویرپردازی رقمه‌ی

فرهاد دلیرانی

۹۶۱۳۱۱۲۵

[dalirani@aut.ac.ir](mailto:dalirani@aut.ac.ir)

[dalirani.1373@gmail.com](mailto:dalirani.1373@gmail.com)

## فهرست

۱	ابزارهای استفاده شده
۲	تمرین ۱
۲	قسمت a
۴	قسمت b
۷	قسمت c
۹	قسمت d
۱۴	قسمت e
۱۶	قسمت f
۱۹	قسمت g
۲۱	قسمت h
۲۴	تمرین ۲
۲۴	قسمت a
۳۲	قسمت b
۴۱	قسمت c
۵۱	قسمت d
۶۰	قسمت e
۶۸	قسمت f
۷۷	قسمت g
۸۰	قسمت h
۸۵	تمرین ۳
۸۵	قسمت a
۸۸	قسمت b
۹۱	قسمت c
۹۵	قسمت d
۱۰۴	تمرین ۴
۱۰۴	قسمت a
۱۰۵	قسمت b
۱۰۶	قسمت c
۱۰۸	تمرین ۵
۱۰۸	قسمت a

١١١ .....	قسمت b
١١٢ .....	قسمت c
١١٢ .....	قسمت d
١١٤ .....	تمرين ٦ .....
١١٤ .....	قسمت a
١١٦ .....	قسمت b
١١٩ .....	قسمت c
١٢٤ .....	قسمت d
١٢٧ .....	قسمت e
١٣١ .....	قسمت f
١٣٦ .....	تمرين ٧ .....
١٣٦ .....	قسمت a
١٣٨ .....	قسمت b
١٤٠ .....	تمرين ٨ .....
١٤٠ .....	قسمت a
١٤٦ .....	قسمت b
١٥١ .....	قسمت c
١٥٥ .....	قسمت d
١٦١ .....	قسمت e
١٦٧ .....	قسمت f
١٧١ .....	قسمت g
١٧٤ .....	قسمت h
١٧٩ .....	قسمت i
١٨٤ .....	قسمت j
١٨٨ .....	تمرين ٩ .....
١٨٨ .....	قسمت a
١٨٨ .....	قسمت b
١٨٨ .....	قسمت c
١٨٨ .....	قسمت d
١٨٨ .....	قسمت e



ابزارهای استفاده شده

زبان برنامه نویسی: متلب

محیط توسعه: MatlabR2016

سیستم عامل: Windows 10

## تمرین ۱

کدهای و نتیجه‌های این سوال در پوشید P1a قرار دارد.

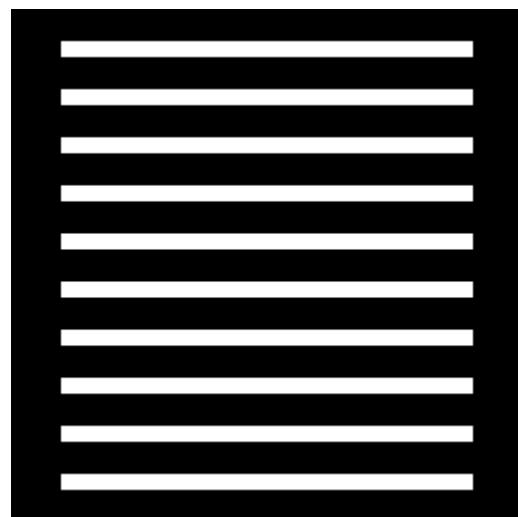
### قسمت a

کد این قسمت در p1a.m قرار دارد. در کد ابتدا تصویر ورودی را می‌خوانم و بعد به ازای سایزهای ۳ در ۳، ۷ در ۷ و ۹ در ۹ فیلتر را اعمال می‌کنم. برای اعمال فیلتر ابتدا تصویر را با مقدار شدت روشنایی صفر pad می‌کنم و Arithmetic سپس به ازای هر پیکسل تصویر ورودی همسایگی آن را که هم اندازه‌ی فیلتر است را پیدا می‌کنم و Mean پیکسل‌های درون همسایگی را بر اساس تابع زیر محاسبه می‌کنم:

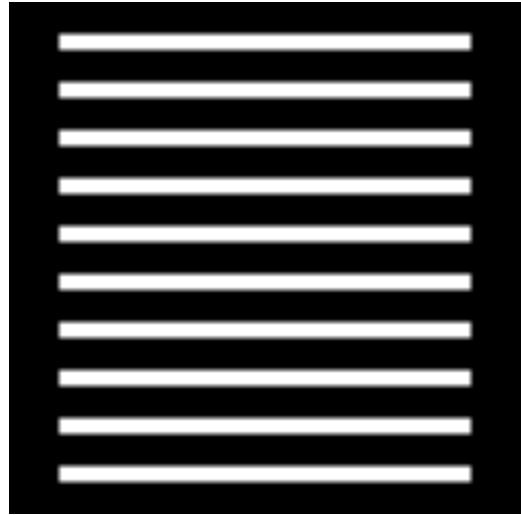
$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s,t)$$

و در ادامه مقدار جدید به دست آمده را جایگزین پیکسل می‌کنم. در زیر خروجی کد را مشاهده می‌کنید:

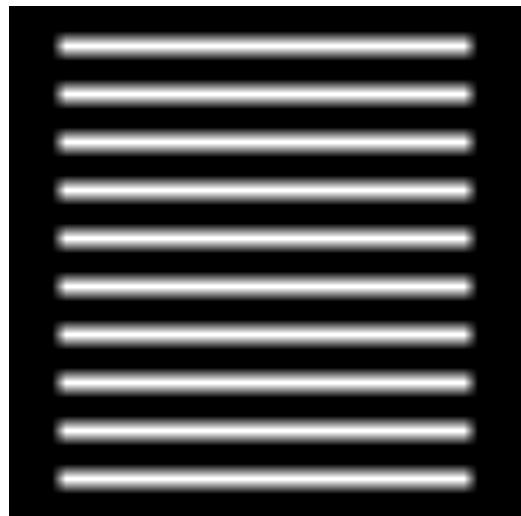
تصویر ورودی:



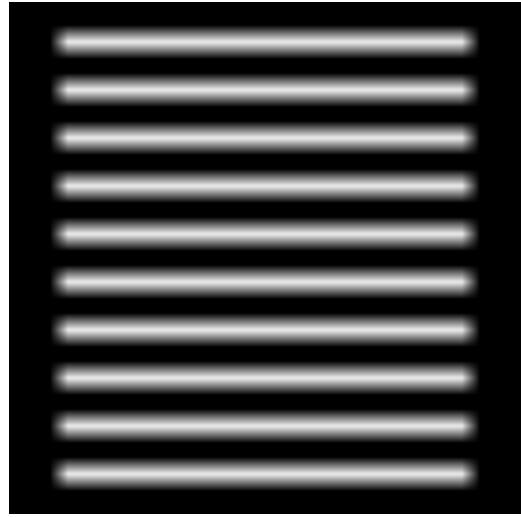
تصویر حاصل از اعمال فیلتر (p1a\_3.png) 3\*3



تصویر حاصل از اعمال فیلتر  $7 \times 7$ : (p1a\_7.png)



تصویر حاصل از اعمال فیلتر  $9 \times 9$ : (p1a\_9.png)



این فیلتر مجموع شدت روشنایی‌های درون همسایگی یک پیکسل را پیدا می‌کند و سپس میانگین آن‌ها را محاسبه می‌کند و در شدت روشنایی پیکسل را با آن آپدیت می‌کند. در نتیجه تصویر بلور می‌شود و در مرز بین سفید و سیاه بلور شدن به خوبی مشاهده می‌شود. هر چه سایز فیلتر بزرگ‌تر شده است میزان بلور شدن افزایش یافته است.

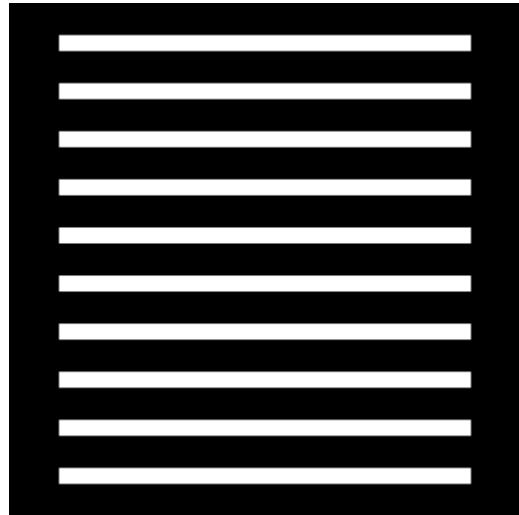
#### قسمت b

کد این قسمت در p1b.m قرار دارد. در کد ابتدا تصویر ورودی را می‌خوانم و بعد به ازای سایزهای ۳ در ۳، ۷ در ۷ و ۹ در ۹ فیلتر را اعمال می‌کنم. برای اعمال فیلتر ابتدا تصویر را با مقدار شدت روشنایی صفر pad می‌کنم و سپس به ازای هر پیکسل تصویر ورودی همسایگی آن را که هم اندازه‌ی فیلتر است را پیدا می‌کنم و **Geometric Mean** پیکسل‌های درون همسایگی را بر اساس تابع زیر محاسبه می‌کنم:

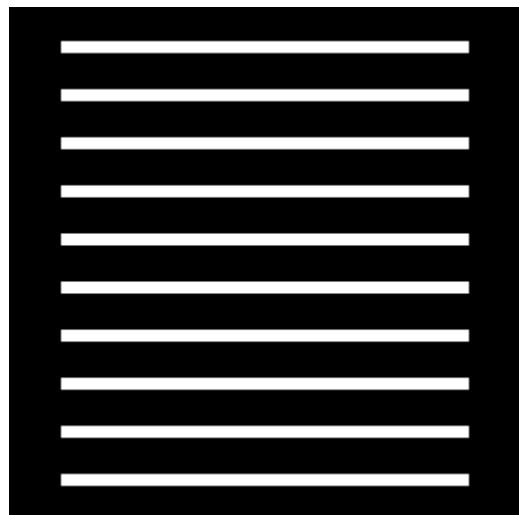
$$\hat{f}(x, y) = \left[ \prod_{(s,t) \in S_{xy}} g(s, t) \right]^{\frac{1}{mn}}$$

و در ادامه مقدار جدید به دست آمده را جایگزین پیسکل می‌کنم. در زیر خروجی کد را مشاهده می‌کنید:

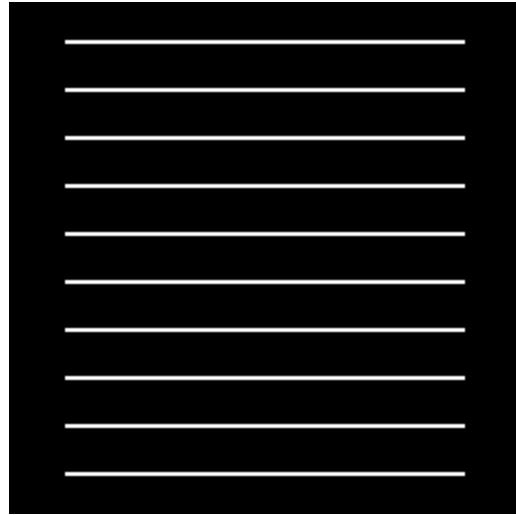
تصویر ورودی:



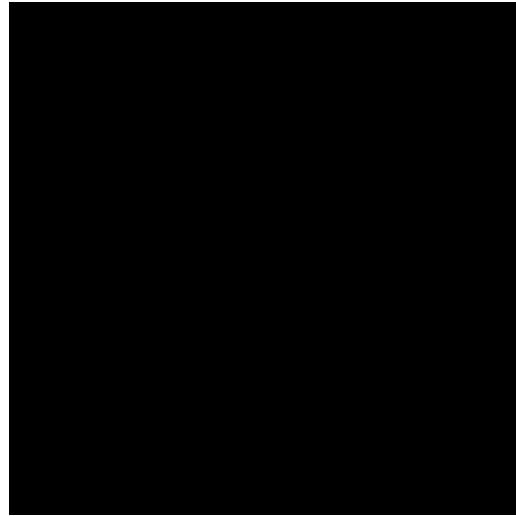
تصویر حاصل از اعمال فیلتر 3\*3 :(p1b\_3.png)



تصویر حاصل از اعمال فیلتر 7\*7 :(p1b\_7.png)



تصویر حاصل از اعمال فیلتر ۹\*۹:(p1b\_9.png)



این فیلتر مجموع شدت روشنایی‌های درون همسایگی یک پیکسل را پیدا می‌کند و سپس ضرب آن‌ها را محاسبه می‌کند و بعد از حاصل ضرب با فرجهی تعداد پیکسل‌ها در همسایگی جزر می‌گیرد و در شدت روشنایی پیکسل را با آن آپدیت می‌کند. از آنجایی که تصویر سیاه و سفید است، اگر در یک همسایگی یک یا بیش از یک پیکسل با شدت روشنایی صفر قرار بگیرد، حاصل ضرب صفر می‌شود و آن نقطه سیاه می‌شود. در غیر آن صورت تمام پیکسل‌ها مقدار ۲۵۵ را دارند که ضرب و جرزشان برابر با ۲۵۵ می‌شود. به همین دلیل بعد از اعمال فیلتر خط‌های سفید باریک‌تر شده‌اند. هر چه اندازه‌ی فیلتر بزرگ‌تر شده است، خط‌های سفید باریک‌تر شده‌اند. به طوری که در فیلتر ۹\*۹ این خط‌ها به کلی حذف شده‌اند.

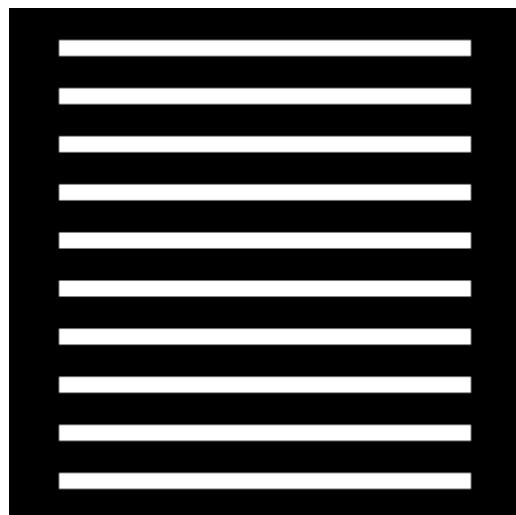
## قسمت ۵

کد این قسمت در p1c.m قرار دارد. در کد ابتدا تصویر ورودی را می‌خوانم و بعد به ازای سایزهای ۳ در ۳، ۷ در ۷ و ۹ در ۹ فیلتر را اعمال می‌کنم. برای اعمال فیلتر ابتدا تصویر را با مقدار شدت روشنایی صفر pad می‌کنم و سپس به ازای هر پیکسل تصویر ورودی همسایگی آن را که هم اندازه‌ی فیلتر است را پیدا می‌کنم و Harmonic Mean پیکسل‌های درون همسایگی را بر اساستابع زیر محاسبه می‌کنم:

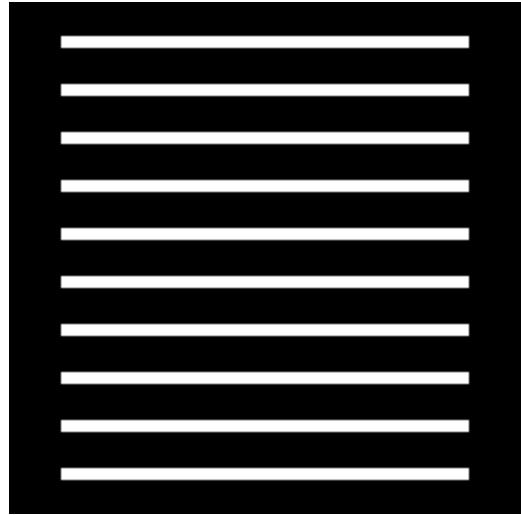
$$\hat{f}(x,y) = \frac{mn}{\sum_{(s,t) \in S_{xy}} \frac{1}{g(s,t)}}$$

و در ادامه مقدار جدید به دست آمده را جایگزین پیکسل می‌کنم. در زیر خروجی کد را مشاهده می‌کنید:

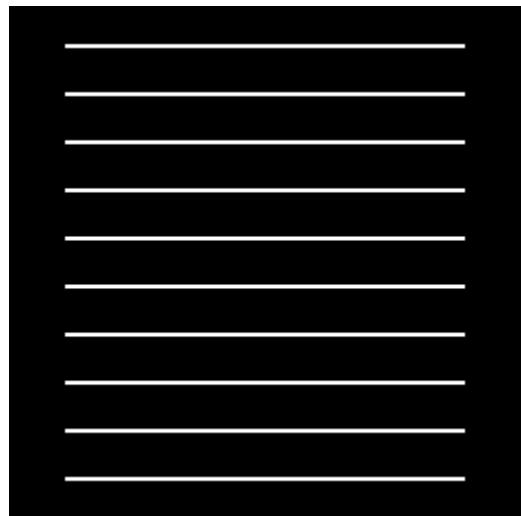
تصویر ورودی:



تصویر حاصل از اعمال فیلتر 3\*3:(p1c\_3.png)



تصویر حاصل از اعمال فیلتر  $7 \times 7$ : p1c\_7.png



تصویر حاصل از اعمال فیلتر  $9 \times 9$ : p1c\_9.png



این فیلتر مجموع معکوس شدت روشنایی‌های درون همسایگی یک پیکسل را پیدا می‌کند و سپس تقسیم تعداد پیکسل‌ها بر آن‌ها را محاسبه می‌کند و شدت روشنایی پیکسل را با آن آپدیت می‌کند. از آنجایی که تصویر سیاه و سفید است، اگر در یک همسایگی یک یا بیش از یک پیکسل با شدت روشنایی صفر قرار بگیرد، حاصل یک تقسیم بر صفر برابر با  $\text{Inf}$  می‌شود و تقسیم نهایی برابر با تعداد پیکسل‌ها بر  $\text{Inf}$  می‌شود که برابر می‌شود با صفر، در نتیجه آن نقطه سیاه می‌شود. در غیر آن صورت تمام پیکسل‌ها مقدار ۲۵۵ را دارند که مقدارشان را در تابع قرار دهیم برابر با ۲۵۵ می‌شود. به همین دلیل بعد از اعمال فیلتر خطاهای سفید باریک‌تر شده‌اند زیرا در مرز سفید و سیاه پیکسل‌های سیاه در همسایگی قرار می‌گیرند. هر چه اندازه‌ی فیلتر بزرگ‌تر شده است، خطاهای سفید باریک‌تر شده‌اند. به طوری که در فیلتر  $9 \times 9$  این خطها به کلی حذف شده‌اند.

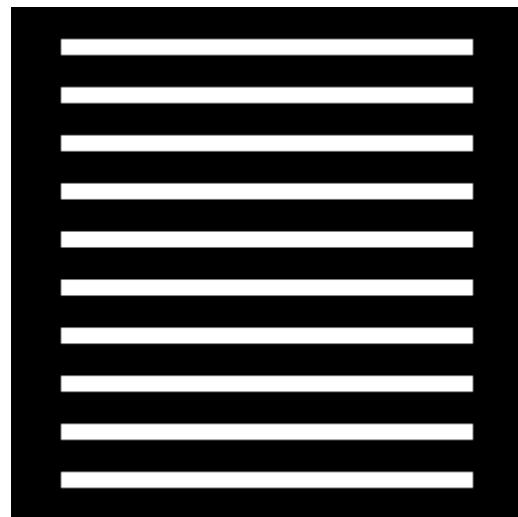
#### قسمت d

کد این قسمت در `p1d.m` قرار دارد. در کد ابتدا تصویر ورودی را می‌خوانم و بعد به ازای سایزهای ۳ در ۳، ۷ در ۷ و ۹ در ۹ فیلتر را اعمال می‌کنم. برای اعمال فیلتر ابتدا تصویر را با مقدار شدت روشنایی صفر `pad` می‌کنم و سپس به ازای هر پیکسل تصویر ورودی همسایگی آن را که هم اندازه‌ی فیلتر است را پیدا می‌کنم و **harmonic Mean** پیکسل‌های درون همسایگی را بر اساس تابع زیر محاسبه می‌کنم:

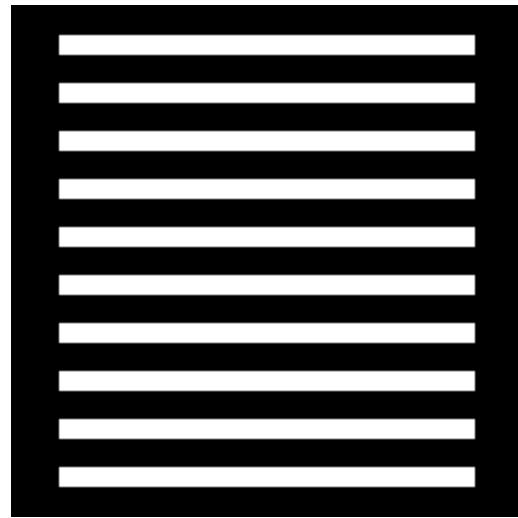
$$\hat{f}(x,y) = \frac{\sum_{(s,t) \in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t) \in S_{xy}} g(s,t)^Q}$$

و در ادامه مقدار جدید به دست آمده را جایگزین پیسکل می‌کنم. در زیر خروجی کد را مشاهده می‌کنید:

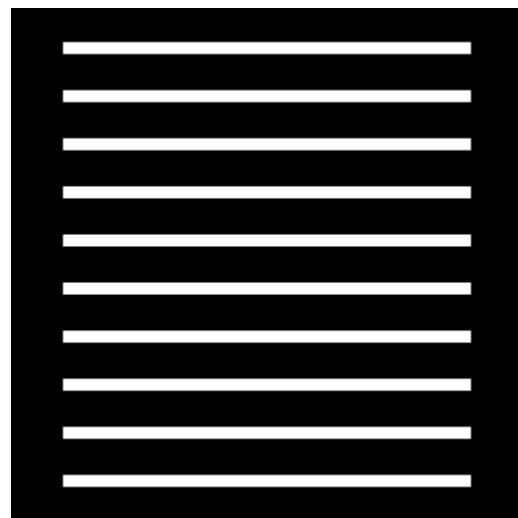
تصویر ورودی:



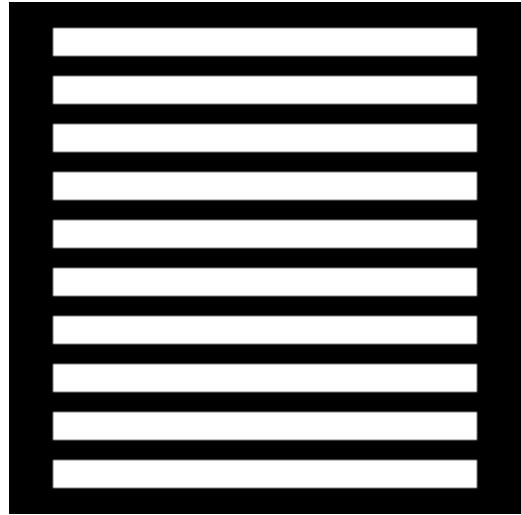
تصویر حاصل از اعمال فیلتر  $3 \times 3$  و  $q=1.5$ : (p1d\_q\_1.5\_3.png)



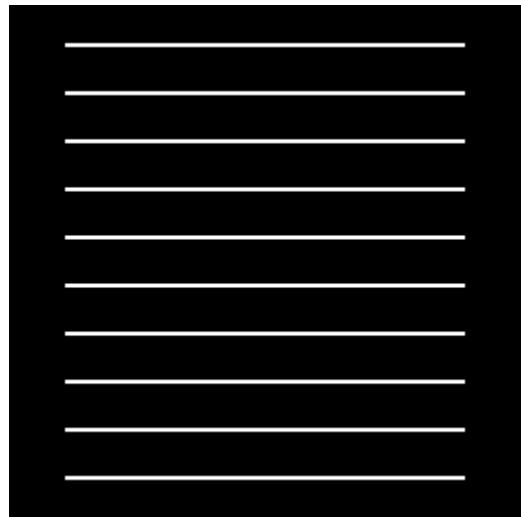
تصویر حاصل از اعمال فیلتر  $3 \times 3$  و  $q=1.5$  (p1d\_q\_-1.5\_3.png)



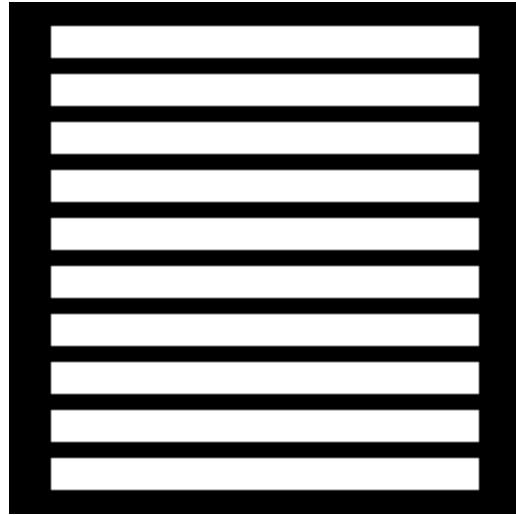
تصویر حاصل از اعمال فیلتر  $7 \times 7$  و  $q=1.5$  (p1d\_q\_1.5\_7.png)



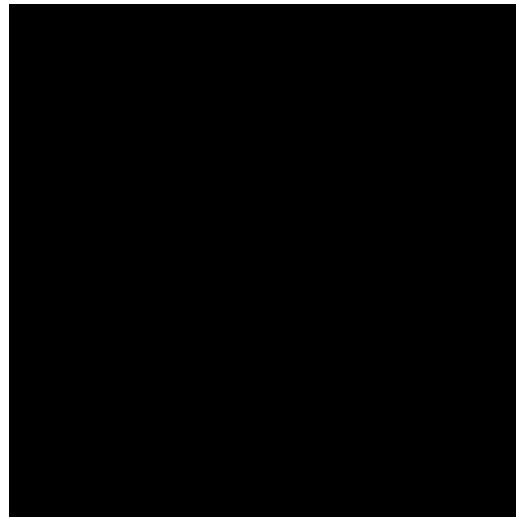
تصویر حاصل از اعمال فیلتر  $7 \times 7$  و  $q=1.5$ : (p1d\_q\_-1.5\_7.png)



تصویر حاصل از اعمال فیلتر  $9 \times 9$  و  $q=1.5$ : (p1d\_q\_1.5\_9.png)



تصویر حاصل از اعمال فیلتر  $9 \times 9$  و  $q=1.5$  : (p1d\_q\_-1.5\_9.png)



این فیلتر مجموع شدت روشنایی‌های همسایگی یک پیکسل به توان  $Q+1$  و مجموع شدت روشنایی‌های همسایگی یک پیکسل به توان  $Q$  را محاسبه می‌کند و دو حاصل جمع را بر هم تقسیم می‌کند و شدت روشنایی پیکسل را با آن آپدیت می‌کند. وقتی  $Q$  برابر با ۱,۵ است، در ناحیه‌ها نزدیک مرز، در همسایگی یک پیکسل، هر دو رنگ سفید و سیاه حضور دارند در نتیجه نوار سفید رنگ بزرگ‌تر می‌شود. هر چه سایز فیلتر بزرگ‌تر شده است نوار سفید هم بزرگ‌تر شده است. وقتی  $Q$  برابر با ۱,۵ است، در صورت وجود صفر در همسایگی، صفر به توان ۱,۵ برابر با بی‌نهایت می‌شود و با فرض بی‌نهایت تقسیم بر بی‌نهایت برابر صفر، در ناحیه‌های مرزی بعد از آپدیت پیکسل سیاه می‌شود به همین دلیل نوار سفید باریک می‌شود. با بزرگ‌شدن فیلتر نوار سفید باریک و باریک‌تر شده است.

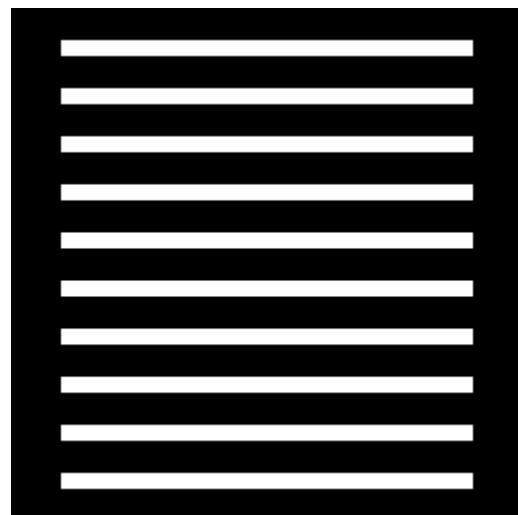
## قسمت e

کد این قسمت در p1e.m قرار دارد. در کد ابتدا تصویر ورودی را می‌خوانم و بعد به ازای سایزهای ۳ در ۳، ۷ در ۷ و ۹ در ۹ فیلتر را اعمال می‌کنم. برای اعمال فیلتر ابتدا تصویر را با مقدار شدت روشنایی صفر pad می‌کنم و سپس به ازای هر پیکسل تصویر ورودی همسایگی آن را که هم اندازهٔ فیلتر است را پیدا می‌کنم و MidPoint Filter پیکسل‌های درون همسایگی را بر اساس تابع زیر محاسبه می‌کنم:

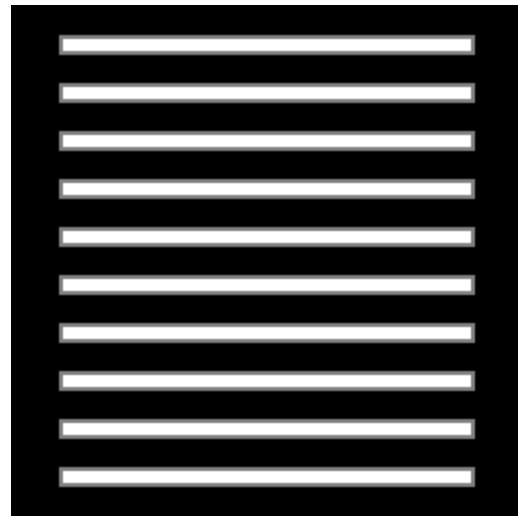
$$\hat{f}(x,y) = \frac{1}{2} \left[ \max_{(s,t) \in S_{xy}} \{g(s,t)\} + \min_{(s,t) \in S_{xy}} \{g(s,t)\} \right]$$

و در ادامه مقدار جدید به دست آمده را جایگزین پیکسل می‌کنم. در زیر خروجی کد را مشاهده می‌کنید:

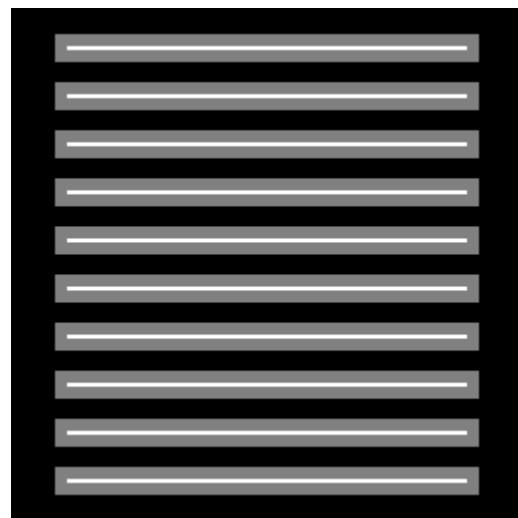
تصویر ورودی:



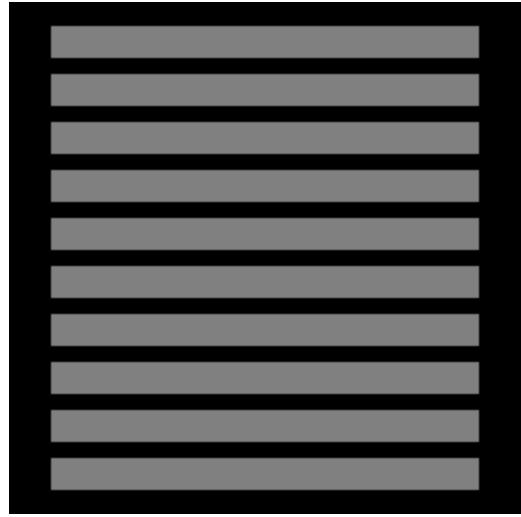
تصویر حاصل از اعمال فیلتر 3\*3 : (p1e\_3.png)



تصویر حاصل از اعمال فیلتر  $7 \times 7$ : p1e\_7.png



تصویر حاصل از اعمال فیلتر  $9 \times 9$ : p1e\_9.png



این فیلتر بزرگ‌ترین و کوچک‌ترین شدت روشنایی‌های درون همسایگی یک پیکسل را پیدا می‌کند و سپس میانگین آن‌ها را محاسبه می‌کند و در شدت روشنایی پیکسل را با آن آپدیت می‌کند. در نتیجه در همسایگی‌هایی که مینیمم و ماکزیمم برابر هم هستند تغییری مشاهده نمی‌شود. ولی در ناحیه‌های نزدیک به مرز، در همسایگی‌ها شاهد حضور پیکسل‌های سفید و سیاه هستیم، در نتیجه مینیمم و ماکزیمم آن‌ها برابر با  $0$  و  $255$  می‌شود که میانگین این دو عدد برابر با  $128$  می‌شود. به همین دلیل در ناحیه‌های نزدیک به مرز شاهد نواری خاکستری رنگ هستیم و هر چه سایز فیلتر بزرگ‌تر شده است این نوار خاکستری رنگ هم بزرگ‌تر شده است.

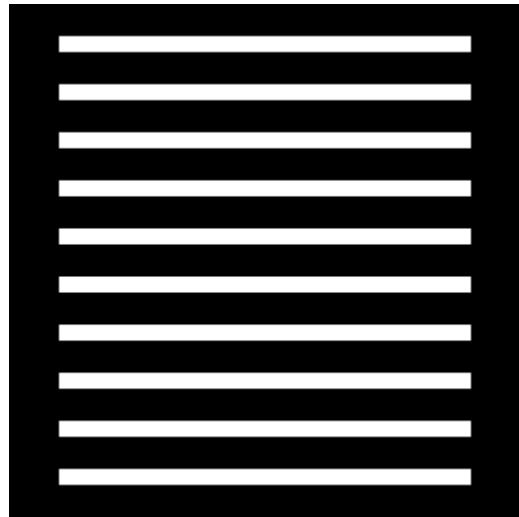
#### قسمت f

کد این قسمت در  $p1f.m$  قرار دارد. در کد ابتدا تصویر ورودی را می‌خوانم و بعد به ازای سایزهای  $3 \times 3$  در  $7 \times 7$  در  $9$  فیلتر را اعمال می‌کنم. برای اعمال فیلتر ابتدا تصویر را با مقدار شدت روشنایی صفر  $pad$  می‌کنم و سپس به ازای هر پیکسل تصویر ورودی همسایگی آن را که هم اندازه‌ی فیلتر است را پیدا می‌کنم و  $Filter$  پیکسل‌های درون همسایگی را بر اساس تابع زیر محاسبه می‌کنم:

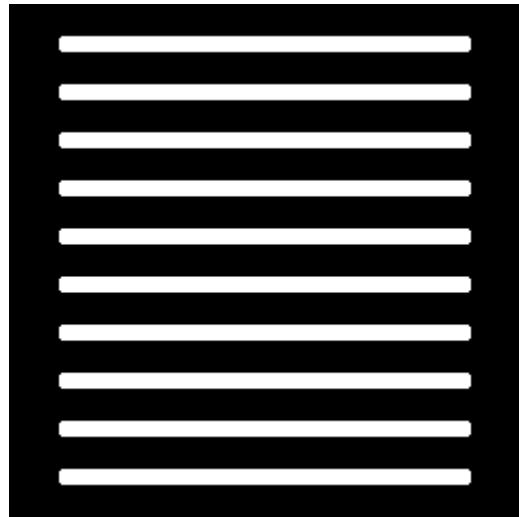
$$\hat{f}(x, y) = \underset{(s,t) \in S_{xy}}{\text{median}} \{g(s, t)\}$$

و در ادامه مقدار جدید به دست آمده را جایگزین پیسکل می‌کنم. در زیر خروجی کد را مشاهده می‌کنید:

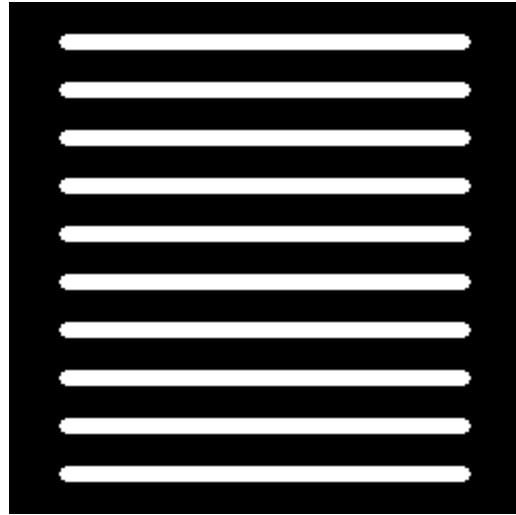
تصویر ورودی:



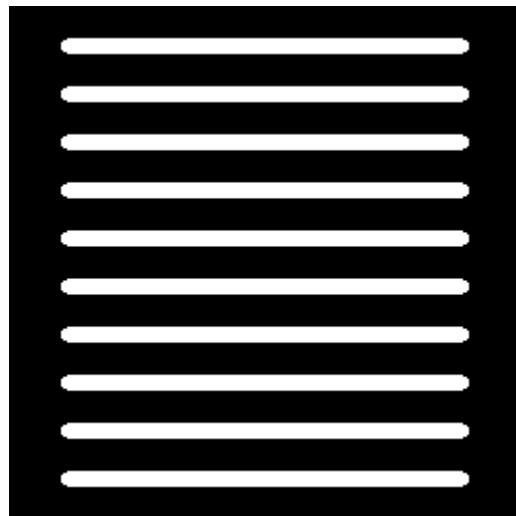
تصویر حاصل از اعمال فیلتر  $3 \times 3$ : (p1f\_3.png)



تصویر حاصل از اعمال فیلتر  $7 \times 7$ : (p1f\_7.png)



تصویر حاصل از اعمال فیلتر 9\*9 : p1f\_9.png



این فیلتر میانه‌ی همسایگی یک پیکسل را پیدا می‌کند و شدت روشنایی پیکسل را با آن آپدیت می‌کند. در همسایگی‌هایی که فقط شامل پیکسل‌های سیاه و یا فقط سفید هستند، مرکز همسایگی همان رنگ خودش را حفظ می‌کند. از آنجایی که سایز فیلترها فرد است، در مرزهای غیر از گوشه‌های سفید، در صورتی که مرکز یک همسایگی سفید باشد، میانه رنگ سفید است و در صورتی که مرکز همسایگی در قسمت سیاه باشد، بیشتر پیکسل‌ها شدت روشنایی سیاه(صفر) دارند و میانه سیاه است. تفاوت فقط در همسایگی‌های اطراف گوشه‌ها رخ می‌دهد به همین دلیل گوشه‌ها گرد می‌شوند. هر چه سایز فیلتر بیشتر شده است، گوشه‌ها بیشتر گرد شده‌اند.

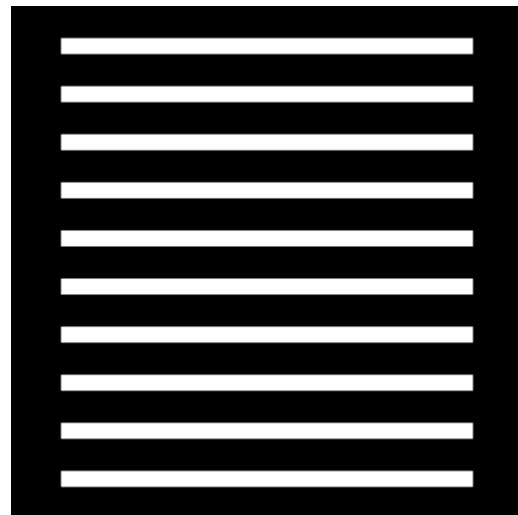
## قسمت ۸

کد این قسمت در p1g.m قرار دارد. در کد ابتدا تصویر ورودی را می‌خوانم و بعد به ازای سایزهای ۳ در ۳، ۷ در ۷ و ۹ در ۹ فیلتر را اعمال می‌کنم. برای اعمال فیلتر ابتدا تصویر را با مقدار شدت روشنایی صفر pad می‌کنم و Min Filter سپس به ازای هر پیکسل تصویر ورودی همسایگی آن را که هم اندازه‌ی فیلتر است را پیدا می‌کنم و پیکسل‌های درون همسایگی را بر اساس تابع زیر محاسبه می‌کنم:

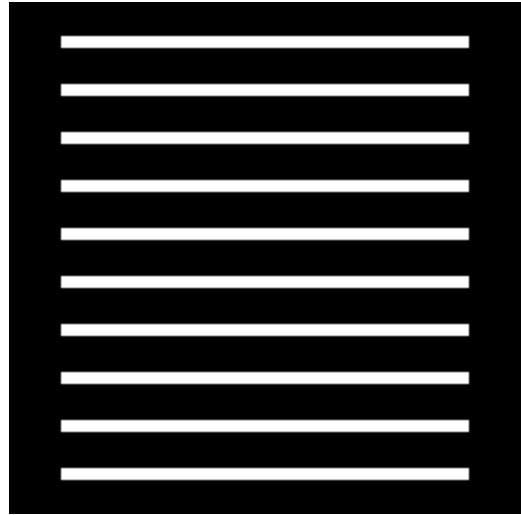
$$\hat{f}(x, y) = \min_{(s,t) \in S_{xy}} \{g(s, t)\}$$

و در ادامه مقدار جدید به دست آمده را جایگزین پیکسل می‌کنم. در زیر خروجی کد را مشاهده می‌کنید:

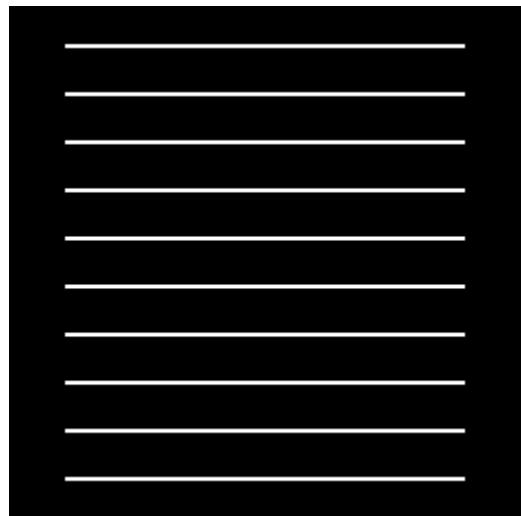
تصویر ورودی:



تصویر حاصل از اعمال فیلتر 3\*3:(p1g\_3.png)



تصویر حاصل از اعمال فیلتر  $7 \times 7$ : p1g\_7.png



تصویر حاصل از اعمال فیلتر  $9 \times 9$ : p1g\_9.png



این فیلتر منیمم همسایگی یک پیکسل را پیدا می‌کند و شدت روشنایی پیکسل را با آن آپدیت می‌کند. در همسایگی‌هایی که فقط شامل پیکسل‌های سیاه و یا فقط سفید هستند، مرکز همسایگی همان رنگ خودش را حفظ می‌کند. در همسایگی‌های اطراف مرز که از دو شدت روشنایی ۲۵۵ و صفر در همسایگی وجود دارد، منیمم برابر با صفر می‌شود به همین دلیل نوار سفید باریک می‌شود. هر چه سایز فیلتر بزرگ‌تر شده است، نوار سفید باریک‌تر شده است به طوری که در فیلتر  $9 \times 9$  این نوار به کلی از بین رفته است.

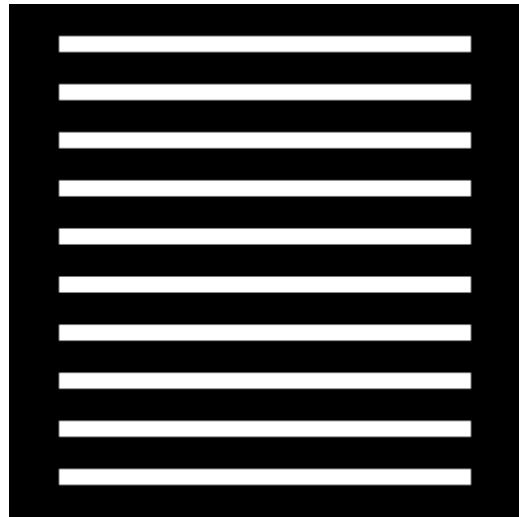
#### قسمت ۱

کد این قسمت در p1h.m قرار دارد. در کد ابتدا تصویر ورودی را می‌خوانم و بعد به ازای سایزهای ۳ در ۳، ۷ در ۷ و ۹ در ۹ فیلتر را اعمال می‌کنم. برای اعمال فیلتر ابتدا تصویر را با مقدار شدت روشنایی صفر pad می‌کنم و سپس به ازای هر پیکسل تصویر ورودی همسایگی آن را که هم اندازه‌ی فیلتر است را پیدا می‌کنم و پیکسل‌های درون همسایگی را بر اساس تابع زیر محاسبه می‌کنم:

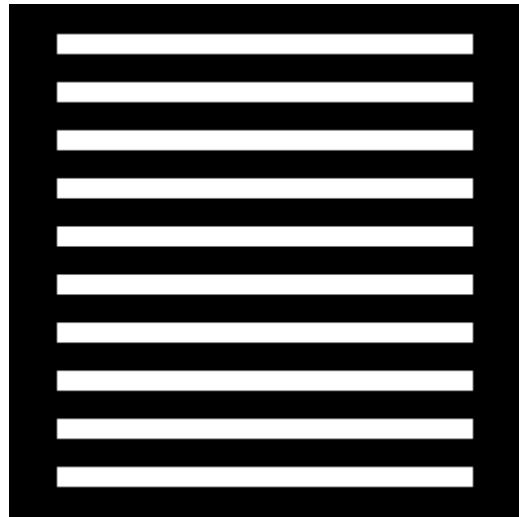
$$\hat{f}(x, y) = \max_{(s,t) \in S_{xy}} \{g(s, t)\}$$

و در ادامه مقدار جدید به دست آمده را جایگزین پیکسل می‌کنم. در زیر خروجی کد را مشاهده می‌کنید:

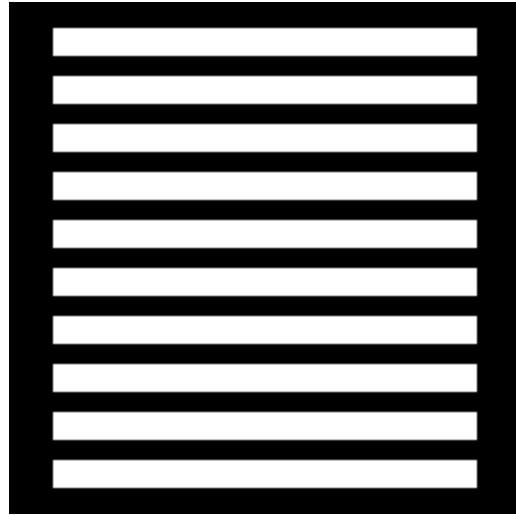
تصویر ورودی:



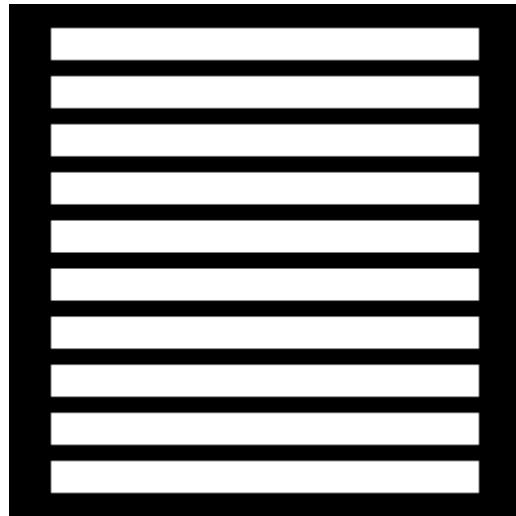
تصویر حاصل از اعمال فیلتر  $3 \times 3$ : (p1h\_3.png)



تصویر حاصل از اعمال فیلتر  $7 \times 7$ : (p1h\_7.png)



تصویر حاصل از اعمال فیلتر  $9 \times 9$ : p1h\_9.png



این فیلتر ماکریم همسایگی یک پیکسل را پیدا می کند و شدت روشنایی پیکسل را با آن آپدیت می کند. در همسایگی هایی که فقط شامل پیکسل های سیاه و یا فقط سفید هستند، مرکز همسایگی همان رنگ خودش را حفظ می کند. در همسایگی های اطراف مرز که از دو شدت روشنایی ۲۵۵ و صفر در همسایگی وجود دارد، ماکریم برابر با ۲۵۵ می شود به همین دلیل نوار سفید رنگ بزرگ تر می شود. هر چه سایز فیلتر بزرگ تر شده است، نوار سفید بزرگ تر شده است.

## تمرین ۲

کدهای این قسمت در پوشه‌ی p2 قرار دارد.

### قسمت a

کدهای این قسمت در p2a.m و p2a\_func.m قرار دارد. در p2a\_func.m تابع زیر قرار دارد:

```
function noisy_img = p2a_func(img, mu, var)
    % This function adds gaussian noise to image.
    % img is an image with pixels value in range [0,1]
    % mu, mean og gaussian distribution
    % var, variance of gaussian distribution
```

این تابع یک تصویر با دیتایپ double می‌گیرد و با استفاده از نویزی که از توزیع Gaussian استخراج می‌کند، تصویر را با توجه به پارامترهای ورودی نویزی می‌کند. در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم.

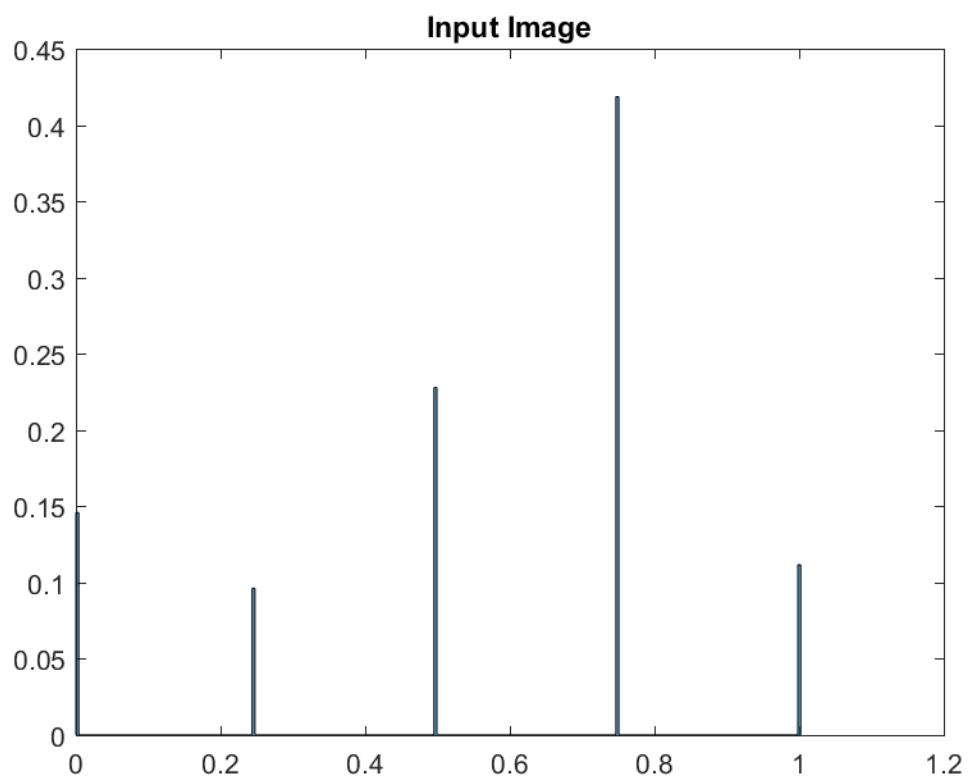
در p2a.m ، تصویر ورودی را از فایل می‌خوانیم سپس هیستوگرام آن را نمایش می‌دهیم و در ادامه با سه مقدار مختلف پارامترهای نویز، تصویر را نویزی می‌کنیم و تصویر نویزی شده و هیستوگرام آن را نمایش می‌دهیم.

در زیر خروجی‌های کد را مشاهده می‌کنید:

تصویر ورودی:



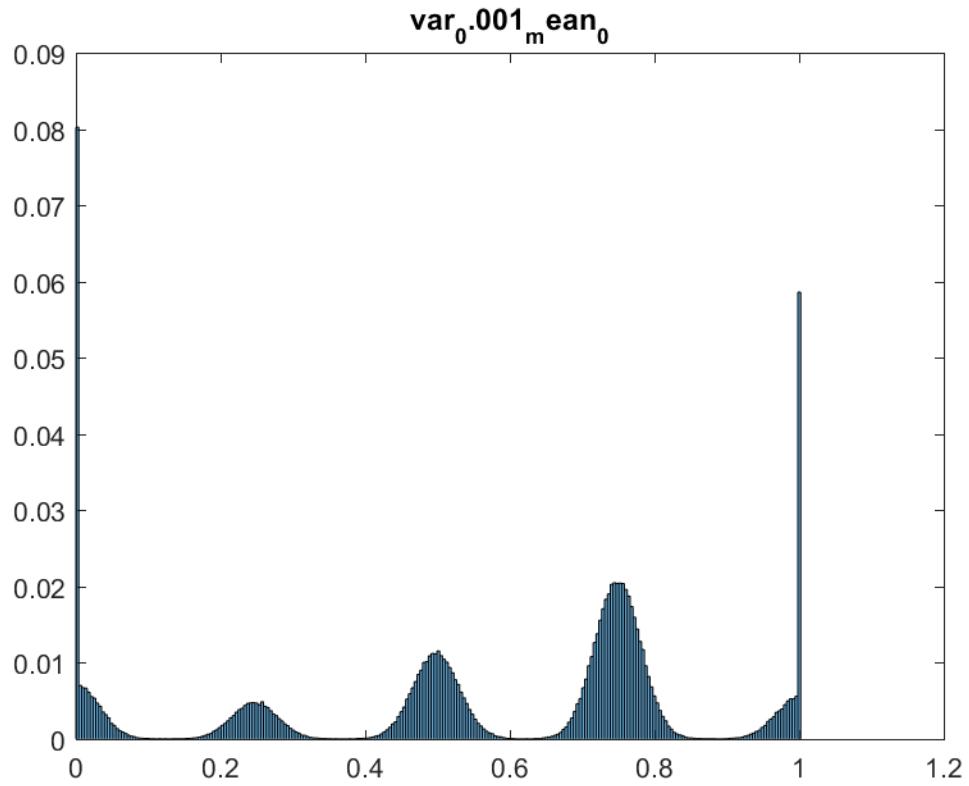
هیستوگرام تصویر ورودی:



تصویر نویزی شده با واریانس ۰،۰۰۱ و میانگین ۰ (p2a\_var\_0.001\_mean\_0.png)



هیستوگرام تصویر نویزی شده با واریانس ۰,۰۰۱ و میانگین ۰  
:(p2a\_hist\_var\_0.001\_mean\_0.png)

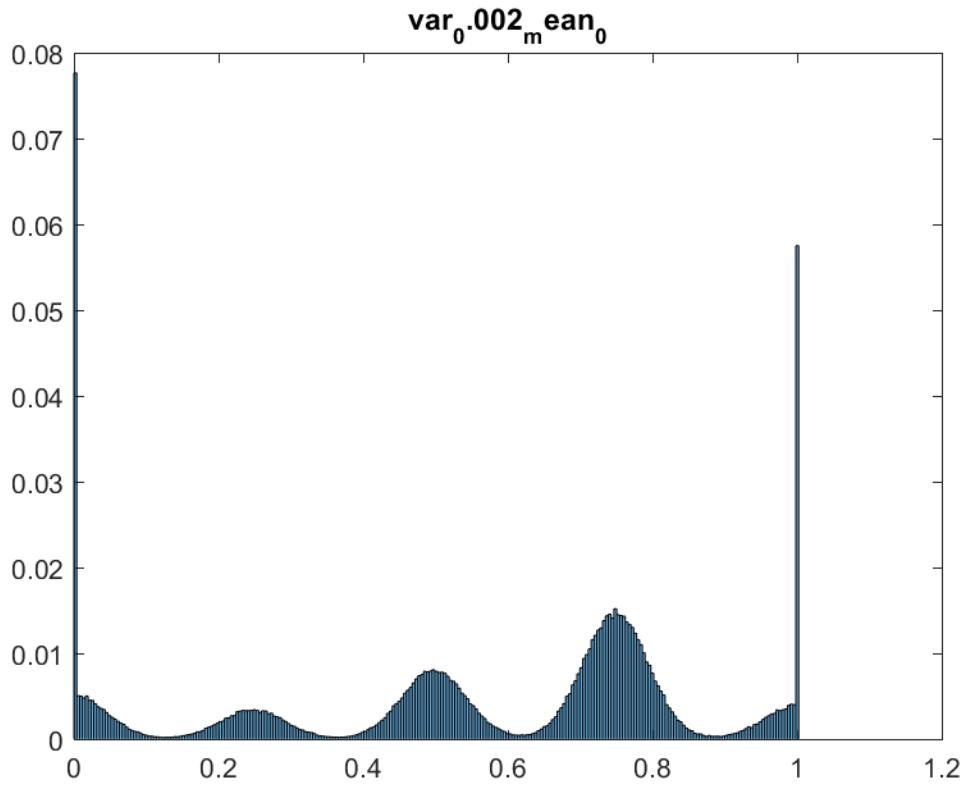


در اطراف ۵ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع گوسی ایجاد شده است، از آنجایی که میانگین نویز صفر قرار داده شده است، محل ۵ شدت روشنایی در هیستوگرام اصلی تغییر نکرده است.

تصویر نویزی شده با واریانس ۰,۰۰۲ و میانگین :



هیستوگرام تصویر نویزی شده با واریانس ۰,۰۰۲ و میانگین ۰  
:(p2a\_hist\_var\_0.002\_mean\_0.png)

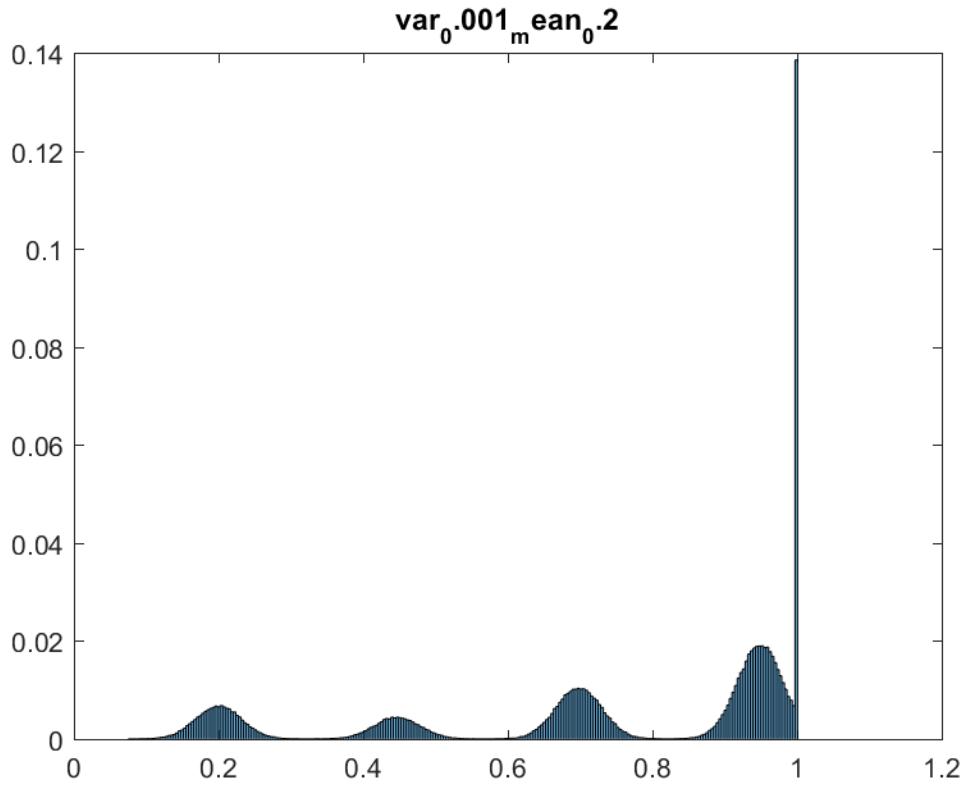


در اطراف ۵ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع گوسی ایجاد شده است، از آنجایی که میانگین نویز صفر قرار داده شده است، محل ۵ شدت روشنایی در هیستوگرام اصلی تغییر نکرده است. در این حالت واریانس نویز نسبت به حالت قبلی بیشتر است در نتیجه همان طور که مشاهده می شود قله ها پهن تر شده اند.

: (p2a\_var\_0.001\_mean\_0.2.png) تصویر نویزی شده با واریانس ۰,۰۰۱ و میانگین ۰,۲



هیستوگرام تصویر نویزی شده با واریانس ۰,۰۰۱ و میانگین ۰,۲  
(p2a\_hist\_var\_0.001\_mean\_0.2.png)



در اطراف شدت روشنایی‌ها در هیستوگرام تصویر بدون نویز، یک توزیع گوسی ایجاد شده است، از آنجایی که میانگین نویز برابر با  $0.2$  قرار داده شده است، محل شدت روشنایی‌ها در هیستوگرام تصویر اصلی به میزان  $0.2$  واحد به سمت راست شیفت داده شده است، به همین دلیل تصویر روشن‌تر به نظر می‌رسد.

در بالا، بعد از هر تصویر نویزی توضیحاتی برای آن‌ها ارائه دادیم. توضیح کلی: در اطراف شدت روشنایی‌های هیستوگرام تصویر ورودی، توزیع گاووسی ایجاد می‌شود. هر چه واریانس بزرگ‌تر می‌شود، قله‌ها کوتاه‌تر می‌شوند و توزیع‌های گوسی پهن‌تر می‌شوند. با تغییر میانگین از صفر به مقدار دیگری، شدت روشنایی‌ها شیفت پیدا می‌کنند.

### قسمت b

کدهای این قسمت در `p2b.m` و `p2b_func.m` قرار دارد. در `p2b_func.m` تابع زیر قرار دارد:

```

function noisy_img = p2b_func(img, pa, pb)
    % This function adds salt and pepper noise to image.
    % img is an image with pixels value in range [0,1]
    % pa, probability of pepper noise
    % Pb, probability of salt noise

```

این تابع یک تصویر با دیتاتایپ **double** می‌گیرد و با استفاده از نویز **Salt and Pepper** تصویر را با توجه به پارامترهای ورودی نویزی می‌کند.

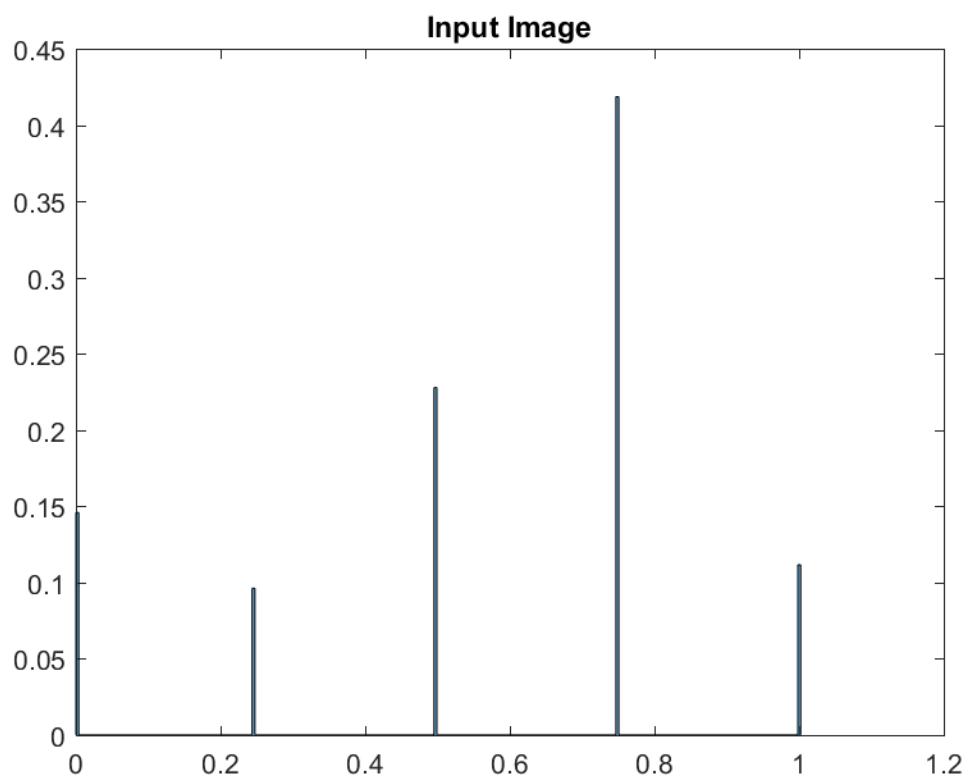
در **p2b.m** ، تصویر ورودی را از فایل می‌خوانیم سپس هیستوگرام آن را نمایش می‌دهیم و در ادامه با سه مقدار مختلف پارامترهای نویز، تصویر را نویزی می‌کنیم و تصویر نویزی شده و هیستوگرام آن را نمایش می‌دهیم.

در زیر خروجی‌های کد را مشاهده می‌کنید:

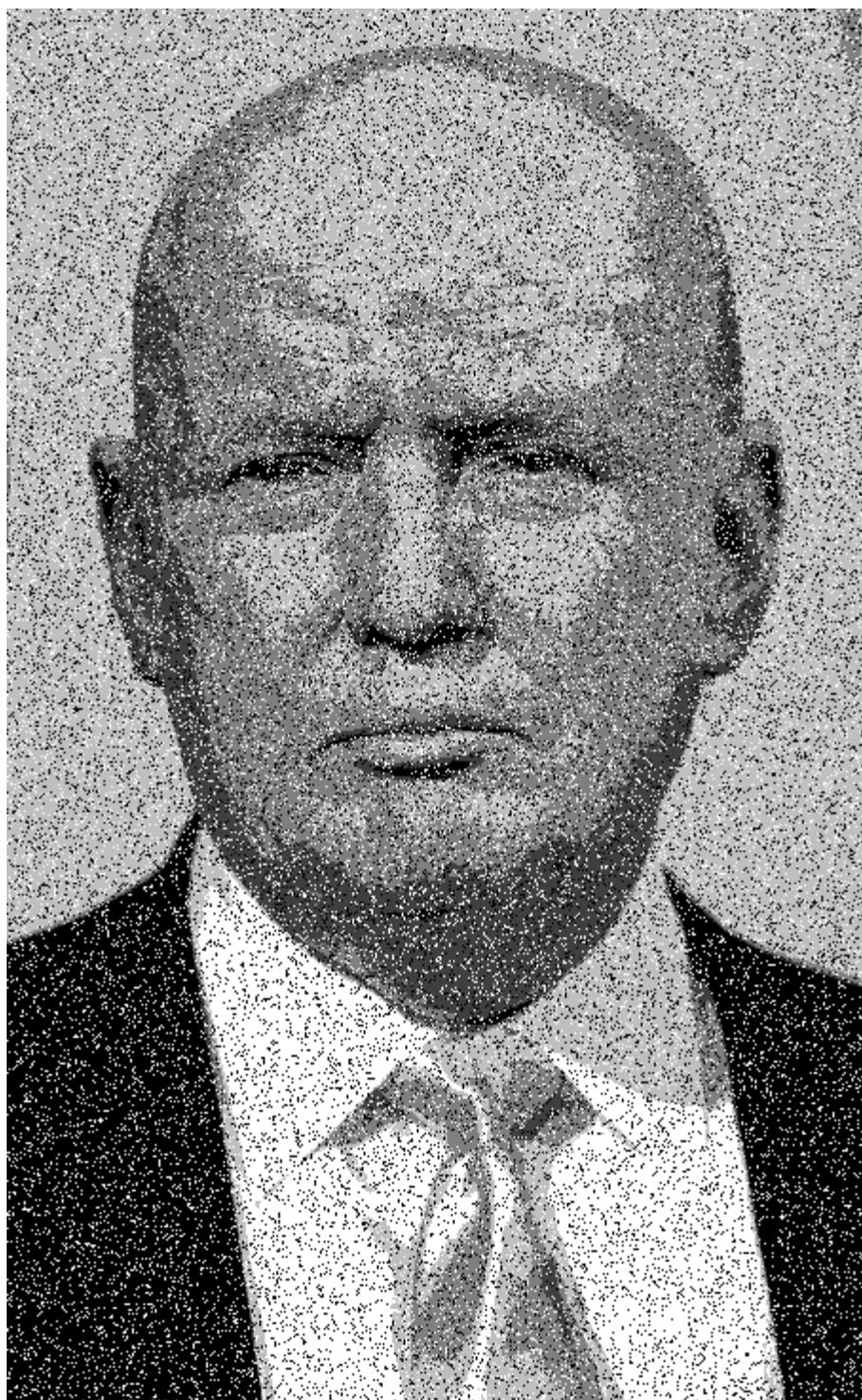
تصویر ورودی:



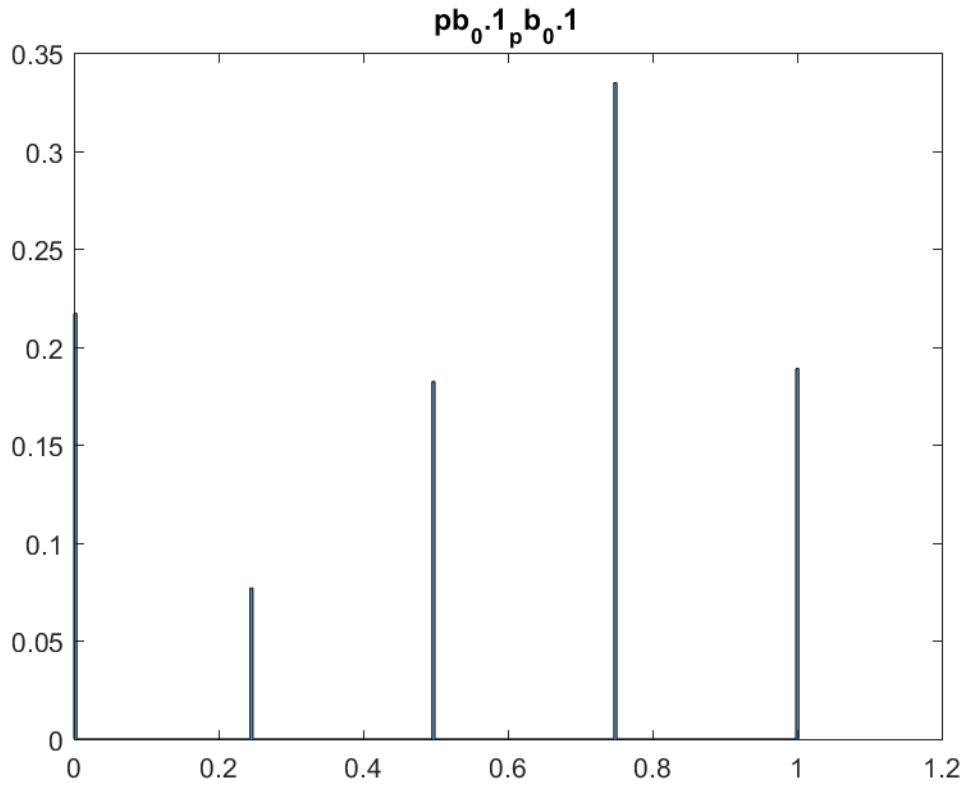
هیستوگرام تصویر ورودی:



تصویر نویزی شده با  $0.1 \text{ Pa}$  و  $0.1 \text{ Pb}$ : (p2b\_pa\_0.1\_pb\_0.1.png)

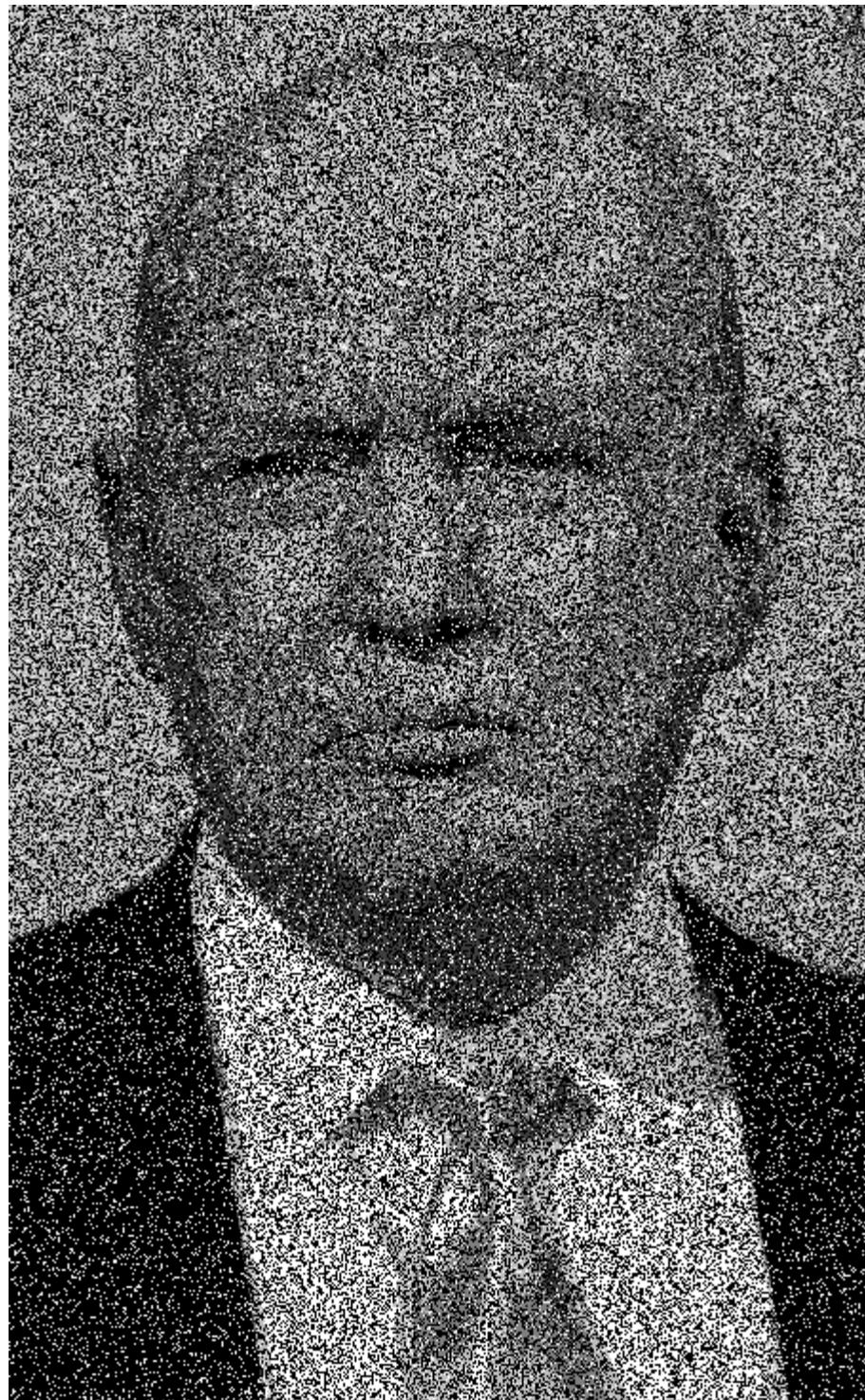


: (p2b\_hist\_pa\_0.1\_pb\_0.1.png) 0.1 Pb و 0.4 Pa هیستوگرام تصویر نویزی شده با

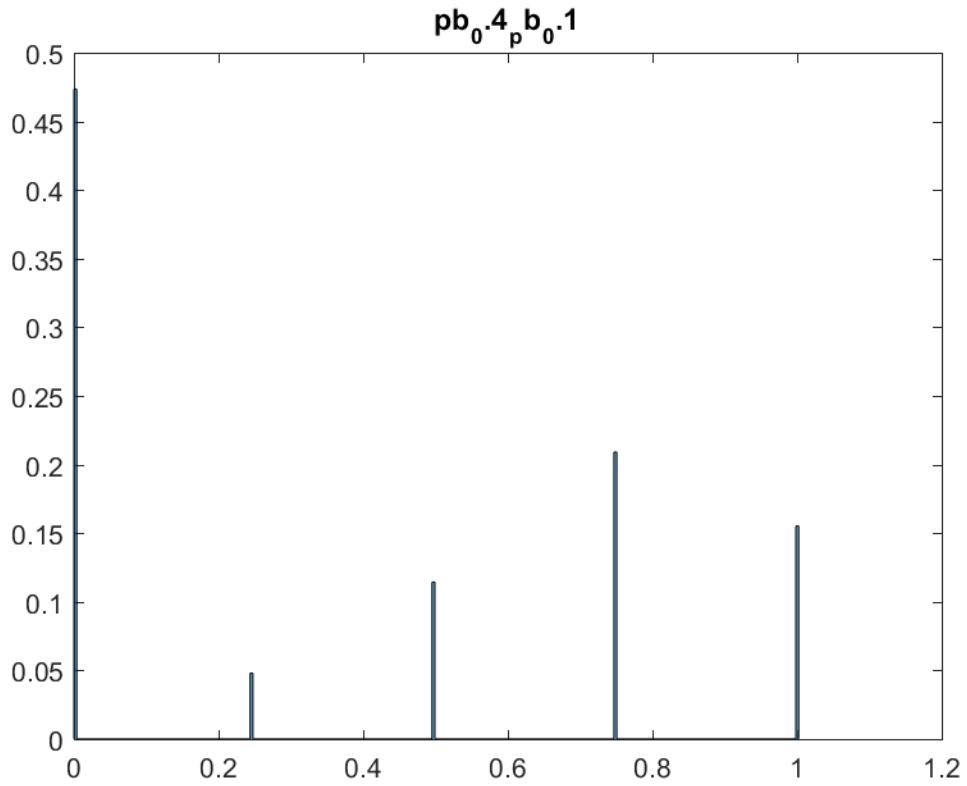


در تصویر بالا احتمال تولید pepper برابر با  $0,1$  است و احتمال تولید salt برابر با  $0,1$  است، به همین دلیل در تصویر نویزی، میزان نویز از هر دو نوع تقریباً یکسان است. از آنجایی که احتمال تولید دو نوع نویز برابر است، در هیستوگرام جدید، میزان شدت روشنایی صفر و یک، تقریباً به یک میزان افزایش داشته است.

تصویر نویزی شده با  $0.1 \text{ Pa}$  و  $0.1 \text{ Pb}$ : (p2b\_pa\_0.4\_pb\_0.1.png)



هیستوگرام تصویر نویزی شده با  $\text{Pa} = 0.4$  و  $\text{Pb} = 0.1$ : (p2b\_hist\_pa\_0.4\_pb\_0.1.png)

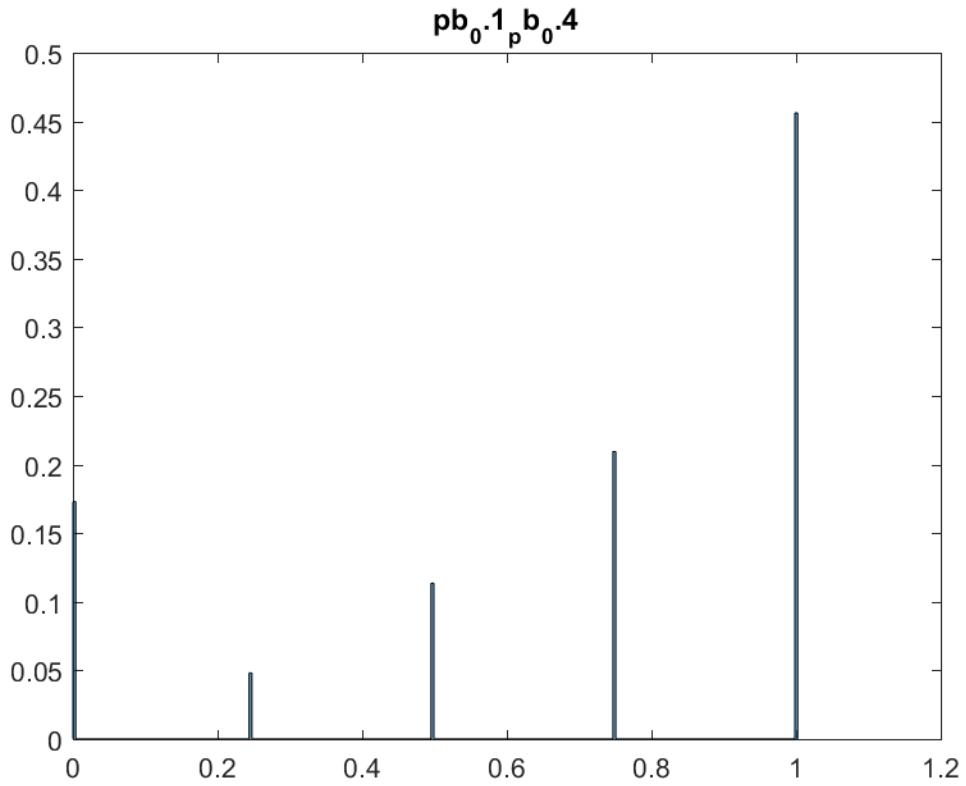


در تصویر بالا احتمال تولید pepper برابر با  $0.4$  است و احتمال تولید salt برابر با  $0.1$  است، به همین دلیل در تصویر نویزی، میزان نویز pepper بیشتر از salt است. از آنجایی که احتمال تولید نویز pepper بیشتر است ، در هیستوگرام جدید، میزان شدت روشنایی صفر افزایش بیشتری نسبت به یک داشته است.

تصویر نویزی شده با  $0.1 \text{ Pa}$  و  $0.4 \text{ Pb}$  :



هیستوگرام تصویر نویزی شده با  $\text{Pa} = 0.1$  و  $\text{Pb} = 0.4$



در تصویر بالا احتمال تولید salt برابر با  $0.4$  است و احتمال تولید pepper برابر با  $0.1$  است، به همین دلیل در تصویر نویزی، میزان نویز salt بیشتر از pepper است. از آنجایی که احتمال تولید نویز salt بیشتر است، در هیستوگرام جدید، میزان شدت روشنایی ۱ افزایش بیشتری نسبت به صفر داشته است.

در بالا، بعد از هر تصویر نویزی توضیحاتی برای آن‌ها ارائه دادیم.

### قسمت C

کدهای این قسمت در p2c\_func.m و p2c.m قرار دارد. در p2c\_func.m تابع زیر قرار دارد:

```
function noisy_img = p2c_func(img, a, b)
    % This function adds uniform noise to image.
    % img is an image with pixels value in range [0,1]
    % a, start of uniform distribution
    % b, end of uniform distribution
```

این تابع یک تصویر با دیتاتایپ `double` می‌گیرد و با استفاده از نویزی که از توزیع Uniform استخراج می‌کند، تصویر را با توجه به پارامترهای ورودی نویزی می‌کند. در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم.

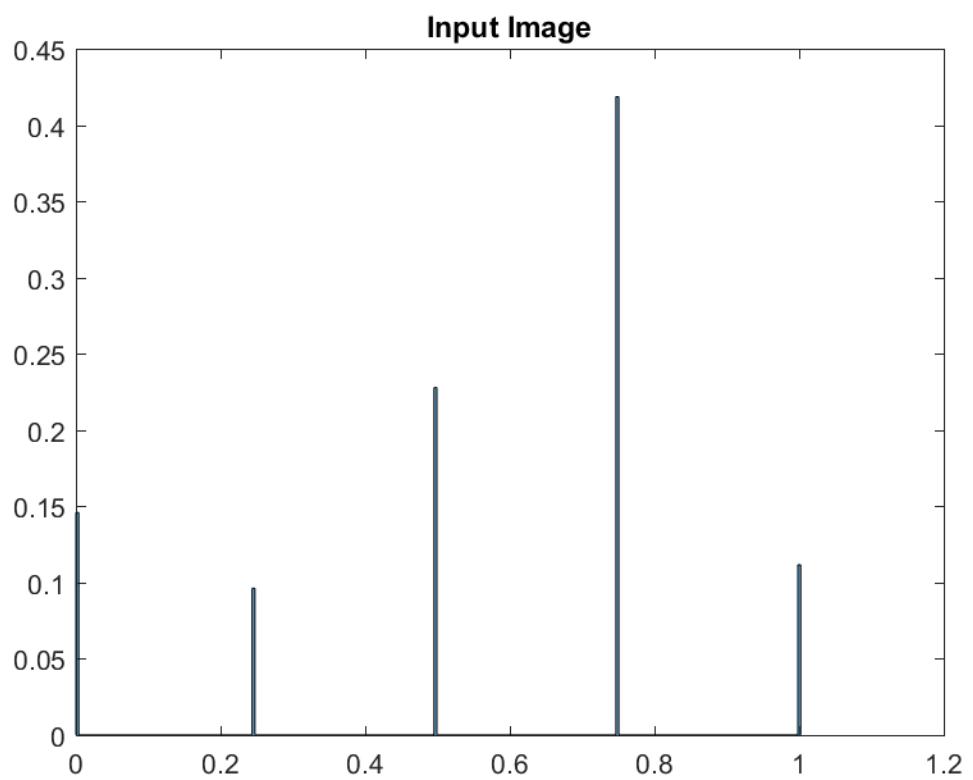
در `p2c.m` ، تصویر ورودی را از فایل می‌خوانیم سپس هیستوگرام آن را نمایش می‌دهیم و در ادامه با سه مقدار مختلف پارامترهای نویز، تصویر را نویزی می‌کنیم و تصویر نویزی شده و هیستوگرام آن را نمایش می‌دهیم.

در زیر خروجی‌های کد را مشاهده می‌کنید:

تصویر ورودی:



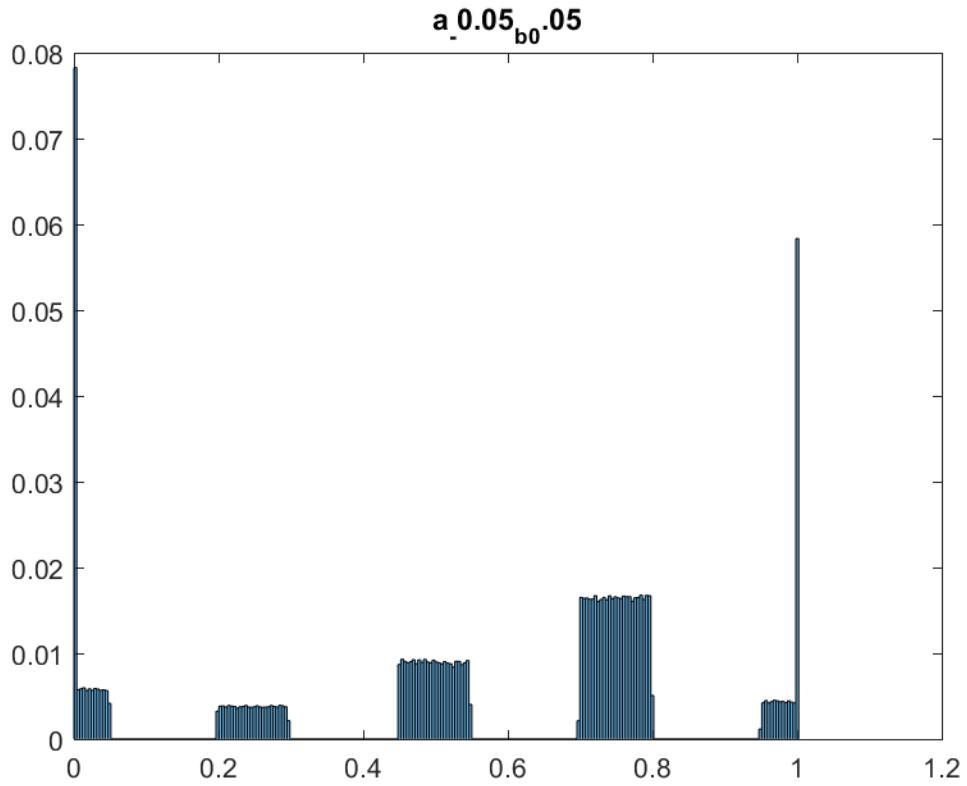
هیستوگرام تصویر ورودی:



: (p2c\_var\_-0.05\_b\_0.05.png)  $0.05 b$  و  $-0.05 a$  تصویر نویزی شده با



: (p2c\_hist\_a\_-0.05\_b\_0.05.png) 0.05 b -0.05 و a نویزی شده با تصویر هیستوگرام



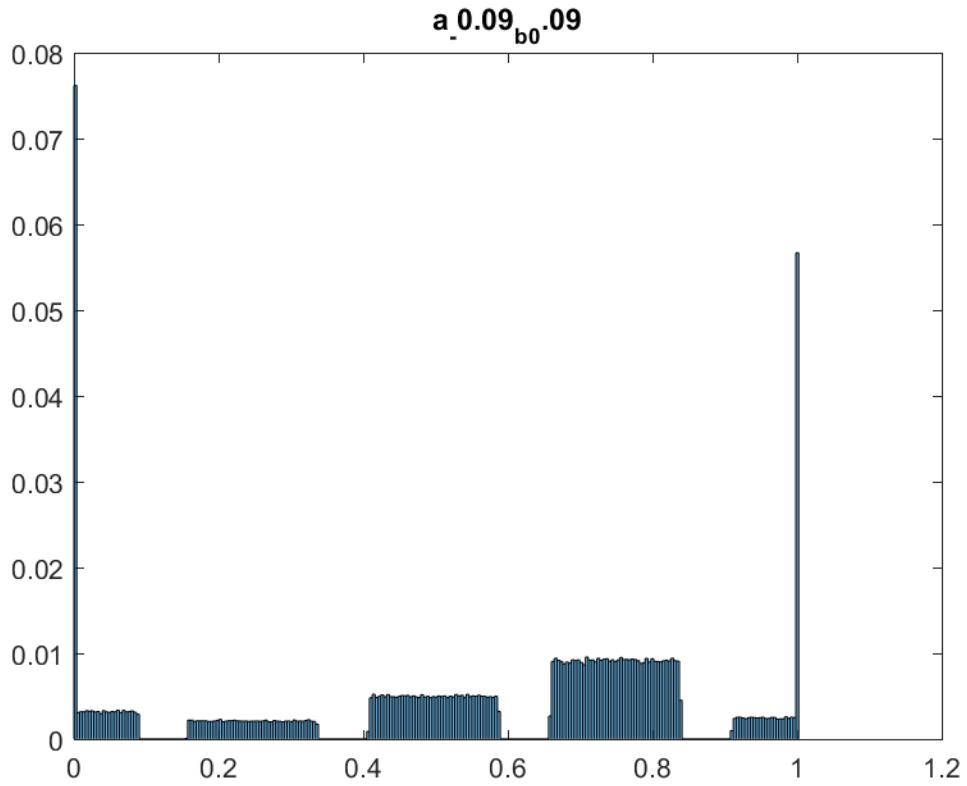
در اطراف ۰ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع یکنواخت ایجاد شده است، از آنجایی که  $a$  و  $b$  منفی یک دیگر قرار داده شده اند، مرکز توزیع‌های یکنواخت برابر است با محل ۰ شدت روشنایی در هیستوگرام تصویر ورودی.

در ابتدا اشاره کردیم که در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم. به همین دلیل در صفر و یک میزان هیستوگرام زیادتر است.

تصویر نویزی شده با  $a = -0.09$  و  $b = 0.09$ : (p2c\_var\_-0.09\_b\_0.09.png)



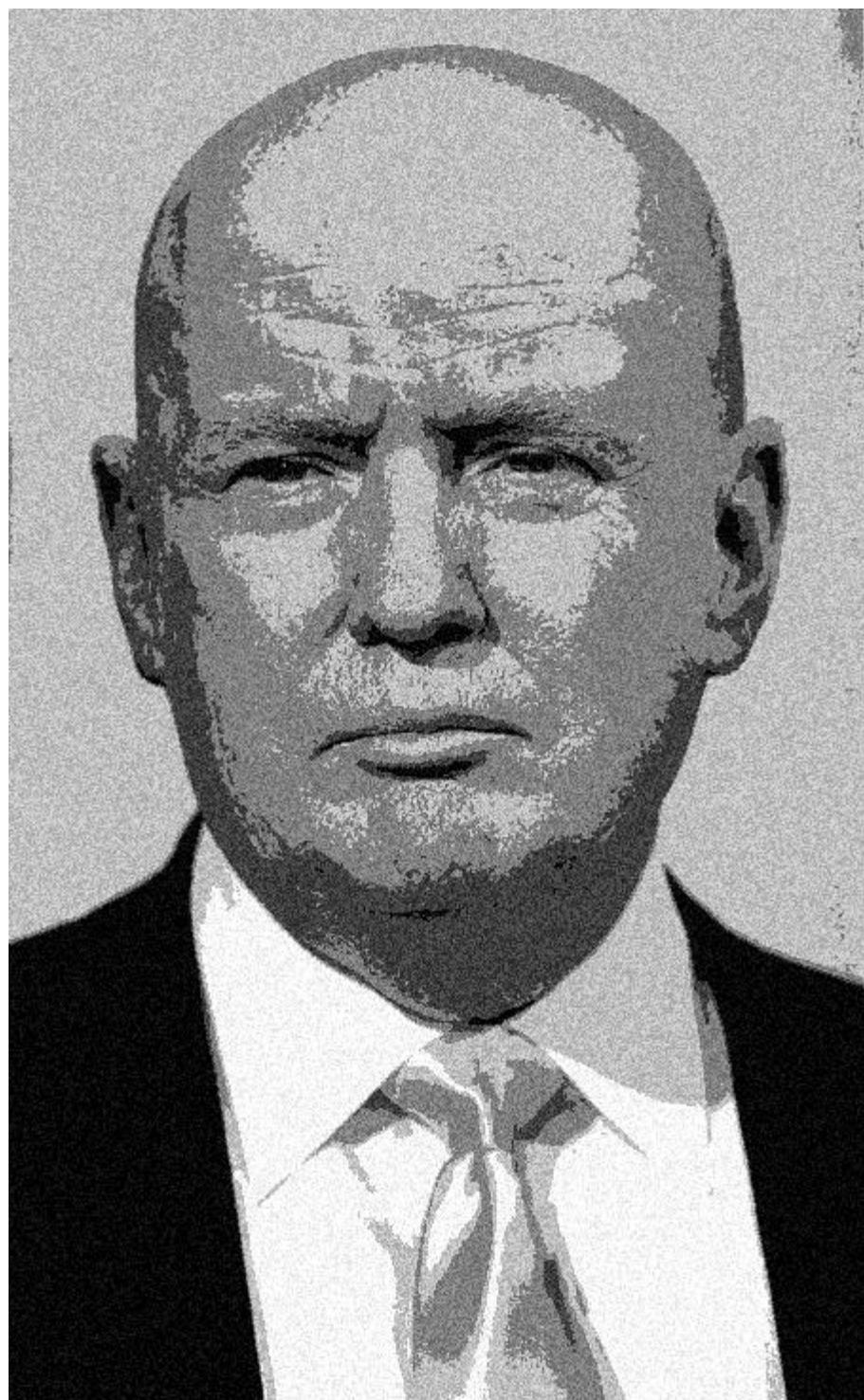
هیستوگرام تصویر نویزی شده با  $a = -0.09$  و  $b = 0.09$ : (p2c\_hist\_a\_-0.09\_b\_0.09.png)



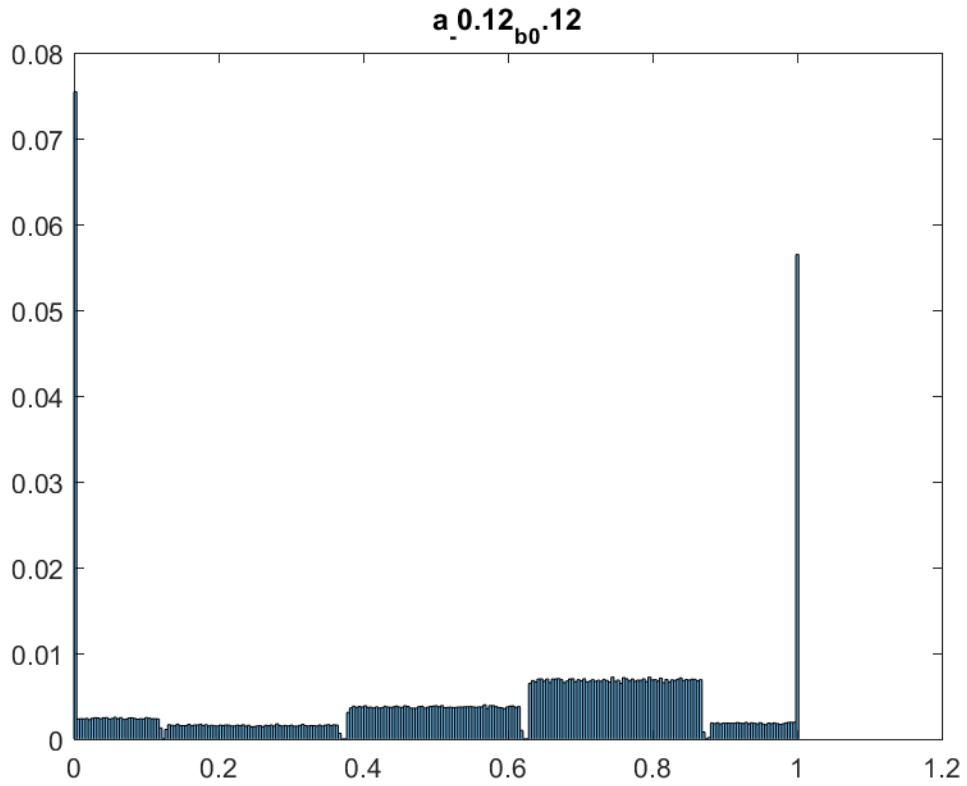
در اطراف ۰ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع یکنواخت ایجاد شده است، از آنجایی که  $a$  و  $b$  منفی یک دیگر قرار داده شده اند، مرکز توزیع‌های یکنواخت برابر است با محل ۰ شدت روشنایی در هیستوگرام تصویر ورودی. در این حالت میزان قدر مطلق  $a$  و  $b$  نسبت به حالت قبل بیشتر است به همین دلیل نویزهای بیشتری در تصویر مشاهده می‌شود و عرض توزیع‌های نرمال در هیستوگرام بیشتر است.

در ابتدا اشاره کردیم که در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم. به همین دلیل در صفر و یک میزان هیستوگرام زیادتر است.

تصویر نویزی شده با  $a = -0.12$  و  $b = 0.12$ : (p2c\_var\_-0.12\_b\_0.12.png)



هیستوگرام تصویر نویزی شده با  $a = -0.12$  و  $b = 0.12$ : (p2c\_hist\_a\_-0.12\_b\_0.12.png)



در اطراف ۰ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع یکنواخت ایجاد شده است، از آنجایی که  $a$  و  $b$  منفی یک دیگر قرار داده شده اند، مرکز توزیع‌های یکنواخت برابر است با محل ۰ شدت روشنایی در هیستوگرام تصویر ورودی. در این حالت میزان قدر مطلق  $a$  و  $b$  نسبت به حالت قبل بیشتر است به همین دلیل نویزهای بیشتری در تصویر مشاهده می‌شود و عرض توزیع‌های نرمال در هیستوگرام بیشتر است.

در ابتدا اشاره کردیم که در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم. به همین دلیل در صفر و یک میزان هیستوگرام زیادتر است.

در بالا، بعد از هر تصویر نویزی توضیحاتی برای آن‌ها ارائه دادیم. توضیح کلی: در اطراف شدت روشنایی‌های هیستوگرام تصویر ورودی، توزیع یک نواحت ایجاد می‌شود. هر چه فاصله‌ی  $a$  و  $b$  بزرگ‌تر می‌شود، نویزی بیشتر در تصویر مشاهده می‌شود و عرض توزیع یکنواخت در هیستوگرام بیشتر می‌شود.

## قسمت d

کدهای این قسمت در p2d.m و p2d\_func.m تابع زیر قرار دارد:

```
function noisy_img = p2d_func(img, a, b)
    % This function adds reyleigh noise to image.
    % img is an image with pixels value in range [0,1]
    % a,
    % b, |
```

این تابع یک تصویر با دیتاتایپ **double** می‌گیرد و با استفاده از نویزی که از توزیع **Reyleigh** استخراج می‌کند، تصویر را با توجه به پارامترهای ورودی نویزی می‌کند. در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم.

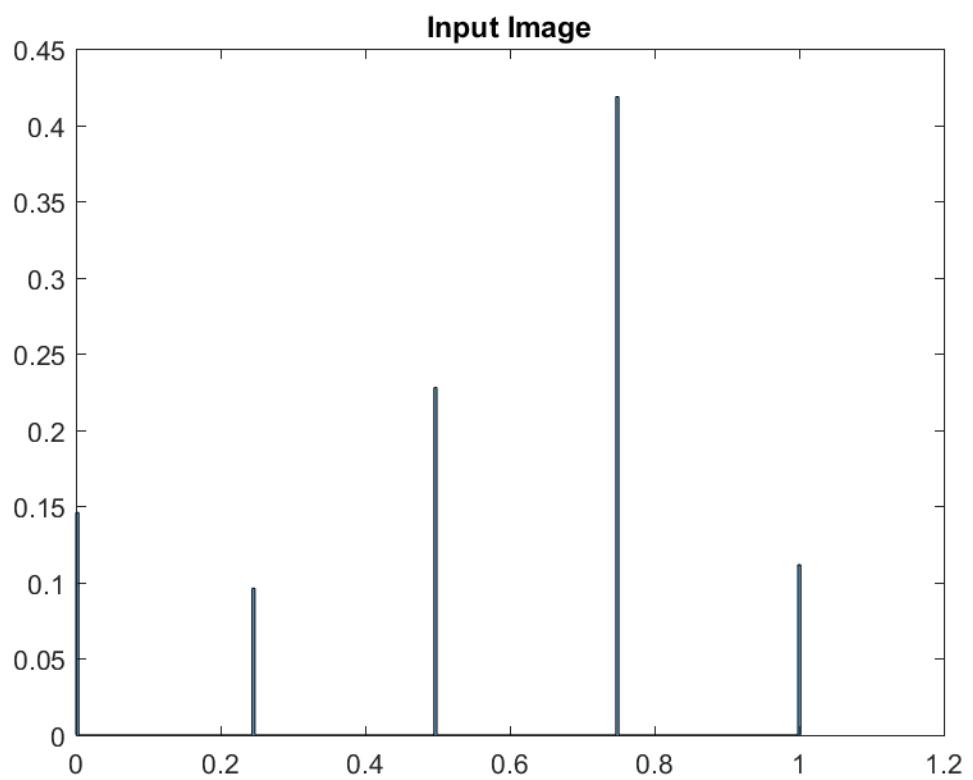
در p2d.m ، تصویر ورودی را از فایل می‌خوانیم سپس هیستوگرام آن را نمایش می‌دهیم و در ادامه با سه مقدار مختلف پارامترهای نویز، تصویر را نویزی می‌کنیم و تصویر نویزی شده و هیستوگرام آن را نمایش می‌دهیم.

در زیر خروجی‌های کد را مشاهده می‌کنید:

تصویر ورودی:



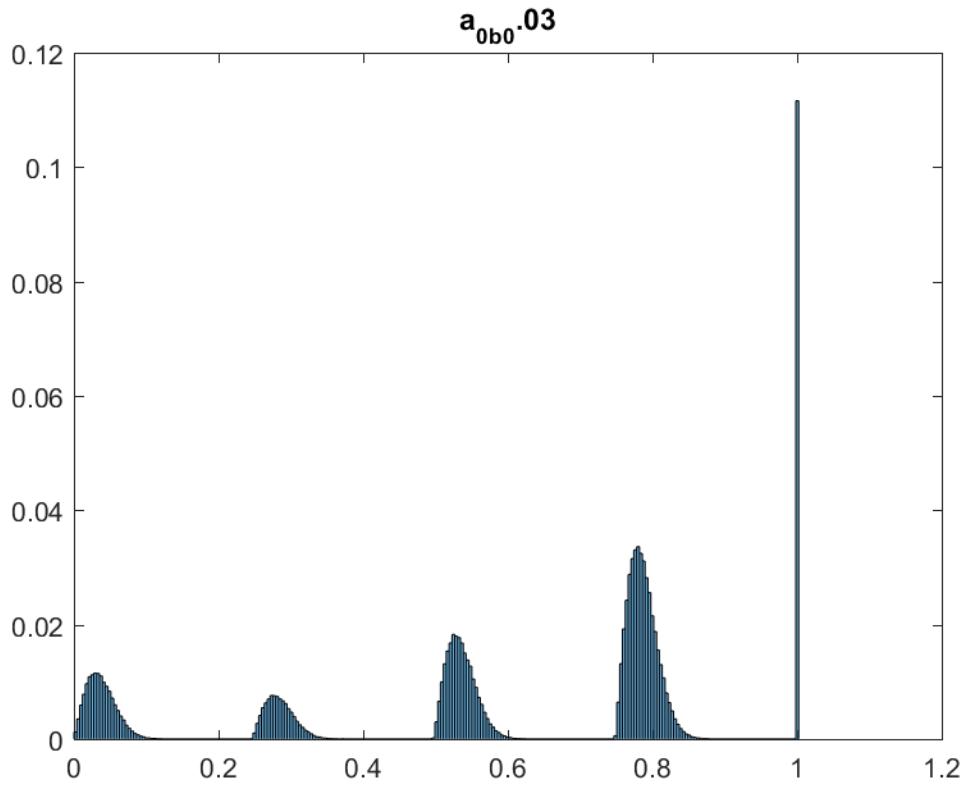
هیستوگرام تصویر ورودی:



تصویر نویزی شده با  $a = 0$  و  $b = 0.03$ : (p2d\_a\_0\_b\_0.03.png)



: (p2d\_hist\_a\_0\_b\_0.03.png) 0.03 b و a با شده نویزی تصویر گرام هیستو



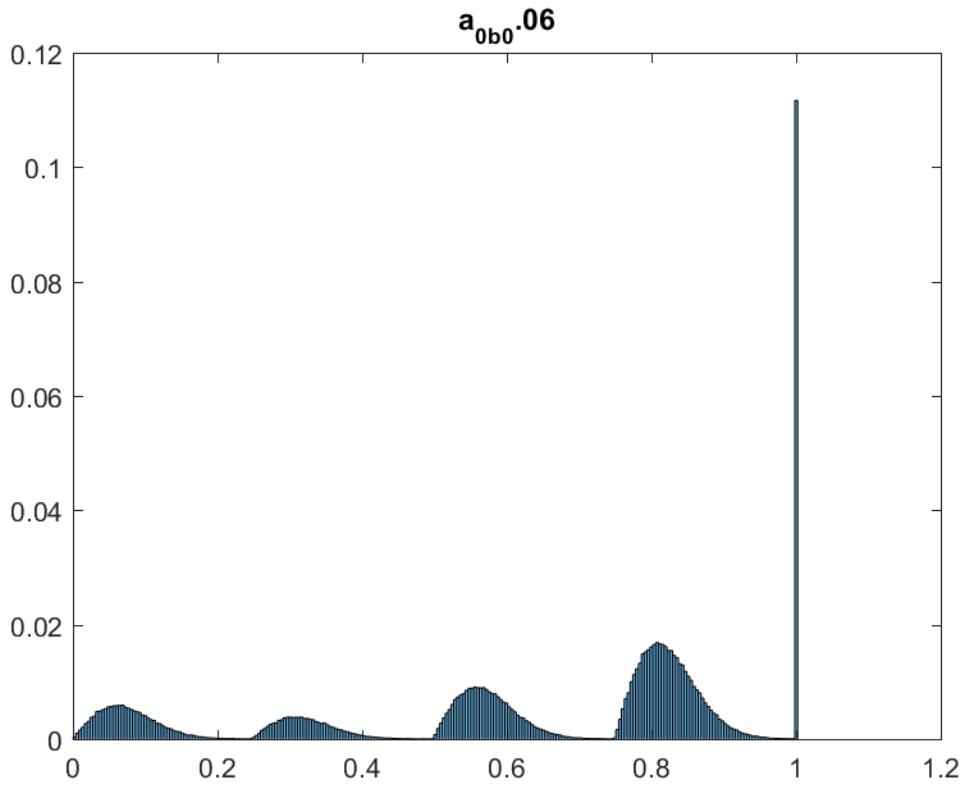
در اطراف ۵ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع Rayleigh ایجاد شده است، از آنجایی که  $a$  برابر با صفر قرار داده شده اند، مرکز توزیع‌های Rayleigh برابر است با محل ۵ شدت روشنایی در هیستوگرام تصویر ورودی.

در ابتدا اشاره کردیم که در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم. به همین دلیل در صفر و یک میزان هیستوگرام زیادتر است.

تصویر نویزی شده با  $a = 0$  و  $b = 0.06$ : (p2d\_a\_0\_b\_0.06.png)



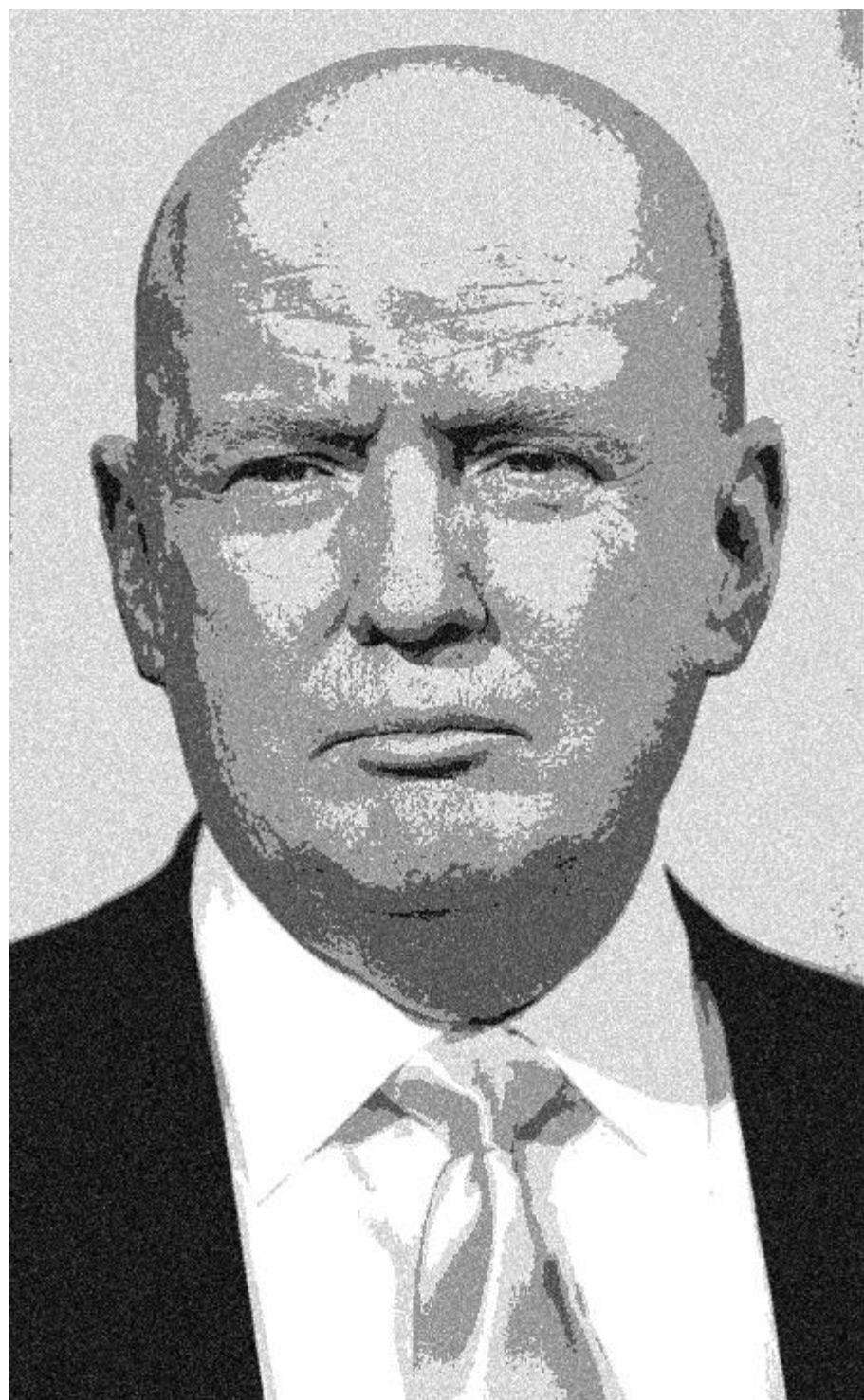
: (p2d\_hist\_a\_0\_b\_0.06.png) 0.06 a و b با شده نویزی تصویر 2D هیستوگرام



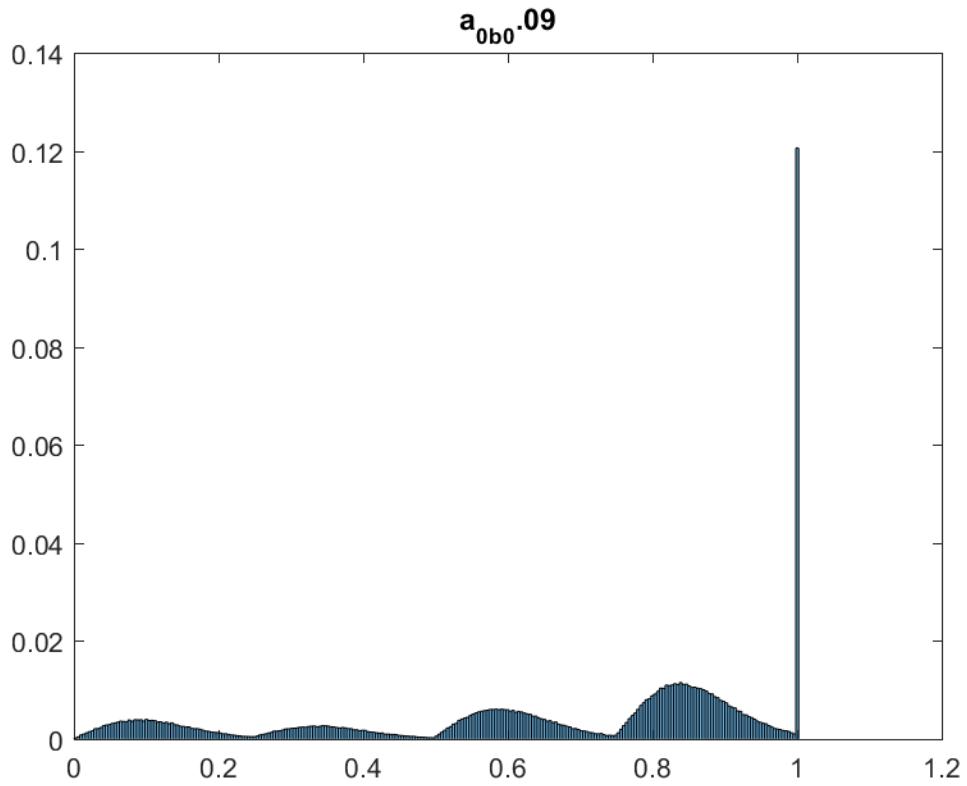
در اطراف ۰ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع Rayleigh ایجاد شده است، از آنجایی که  $a$  برابر با صفر قرار داده شده اند، مرکز توزیع‌های Rayleigh برابر است با محل ۰ شدت روشنایی در هیستوگرام تصویر ورودی. در این حالت نسبت به حالت قبل، میزان  $b$  بیشتر است، به همین دلیل در تصویر نویز بیشتری مشاهده می‌شود و ارتفاع توزیع در هیستوگرام کمتر شده است و توزیع عریض‌تر شده است.

در ابتدا اشاره کردیم که در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم. به همین دلیل در صفر و یک میزان هیستوگرام زیادتر است.

تصویر نویزی شده با  $a = 0$  و  $b = 0.09$ : (p2d\_a\_0\_b\_0.09.png)



: (p2d\_hist\_a\_0\_b\_0.09.png) 0.09 b و a با شده نویزی تصویر گرام هیستو



در اطراف ۵ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع Rayleigh ایجاد شده است، از آنجایی که  $a$  برابر با صفر قرار داده شده اند، مرکز توزیع‌های Rayleigh برابر است با محل ۵ شدت روشنایی در هیستوگرام تصویر ورودی. در این حالت نسبت به حالت قبل، میزان  $b$  بیشتر است، به همین دلیل در تصویر نویز بیشتر مشاهده می‌شود و ارتفاع توزیع در هیستوگرام کمتر شده است و توزیع عریض‌تر شده است.

در ابتدا اشاره کردیم که در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم. به همین دلیل در صفر و یک میزان هیستوگرام زیادتر است.

در بالا، بعد از هر تصویر نویزی توضیحاتی برای آن‌ها ارائه دادیم. توضیح کلی: در اطراف شدت روشنایی‌های هیستوگرام تصویر ورودی، توزیع یک Rayleigh ایجاد می‌شود. هر چه فاصله‌ی  $b$  بزرگ‌تر می‌شود، نویزی بیشتر در تصویر مشاهده می‌شود و عرض توزیع یکنواخت در هیستوگرام بیشتر می‌شود و ارتفاع قله‌ها کاهش می‌یابد.

کدهای این قسمت در p2e.m و p2e\_func.m تابع زیر قرار دارد:

---

```

function noisy_img = p2e_func(img, a, b)
    % This function adds gamma noise to image.
    % img is an image with pixels value in range [0,1]
    % a, shape parameter
    % b, scale parameter
```

این تابع یک تصویر با دیتابایپ **double** می‌گیرد و با استفاده از نویزی که از توزیع Gamma استخراج می‌کند، تصویر را با توجه به پارامترهای ورودی نویزی می‌کند. در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم.

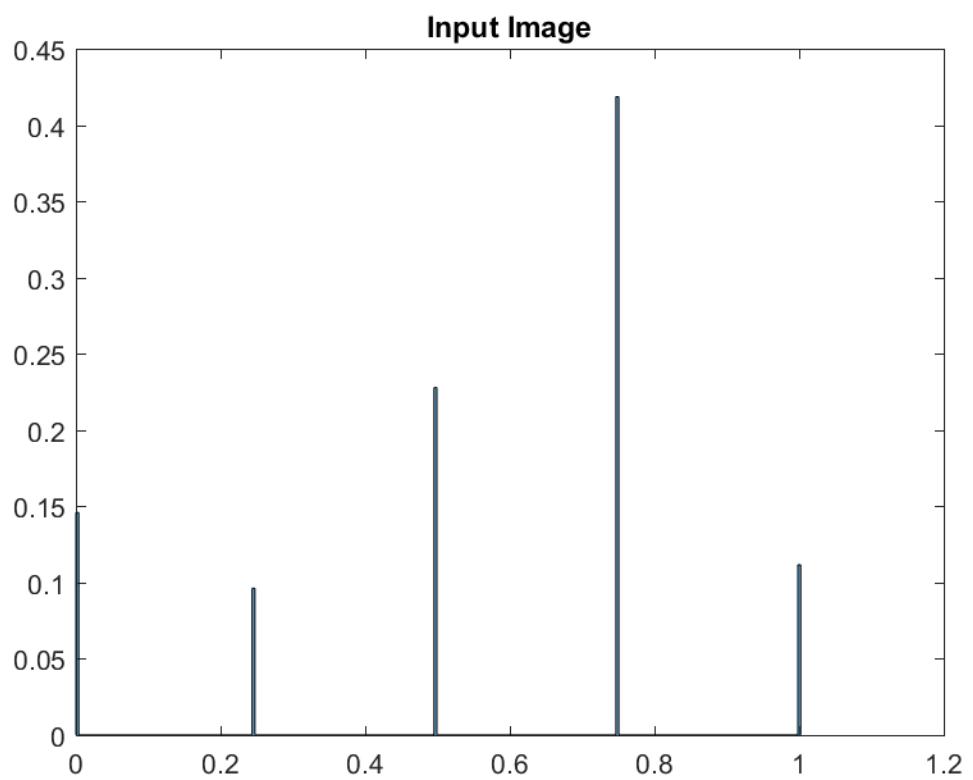
در p2e.m ، تصویر ورودی را از فایل می‌خوانیم سپس هیستوگرام آن را نمایش می‌دهیم و در ادامه با سه مقدار مختلف پارامترهای نویز، تصویر را نویزی می‌کنیم و تصویر نویزی شده و هیستوگرام آن را نمایش می‌دهیم.

در زیر خروجی‌های کد را مشاهده می‌کنید:

تصویر ورودی:



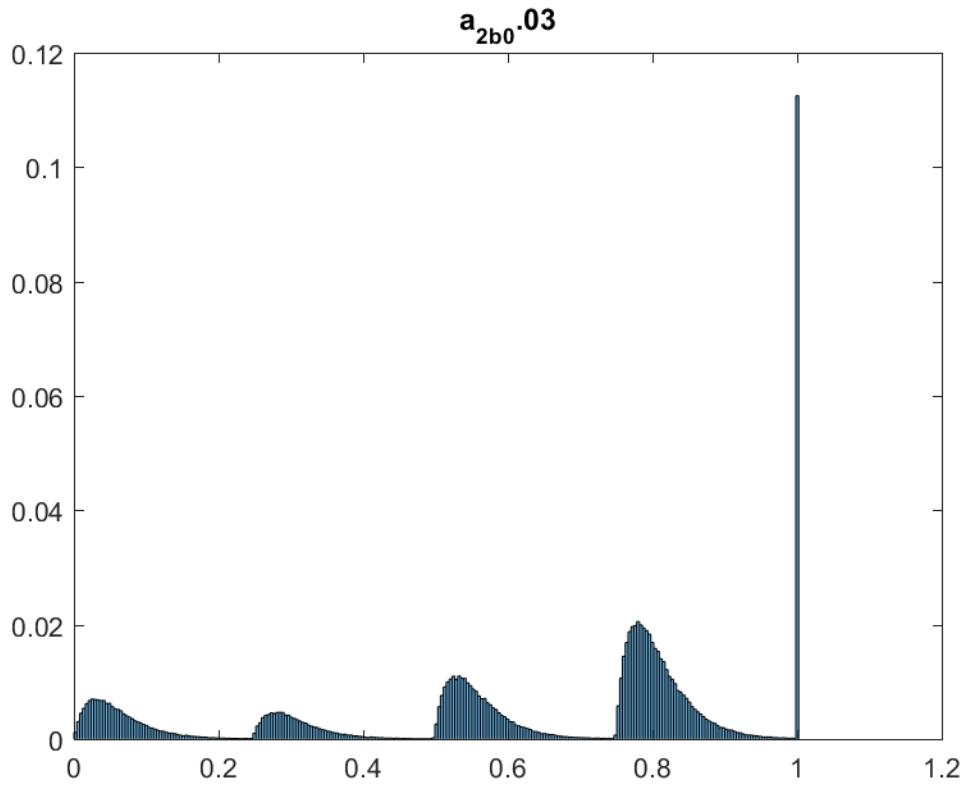
هیستوگرام تصویر ورودی:



تصویر نویزی شده با  $a = 2$  و  $b = 0.03$ : (p2e\_a\_2\_b\_0.03.png)



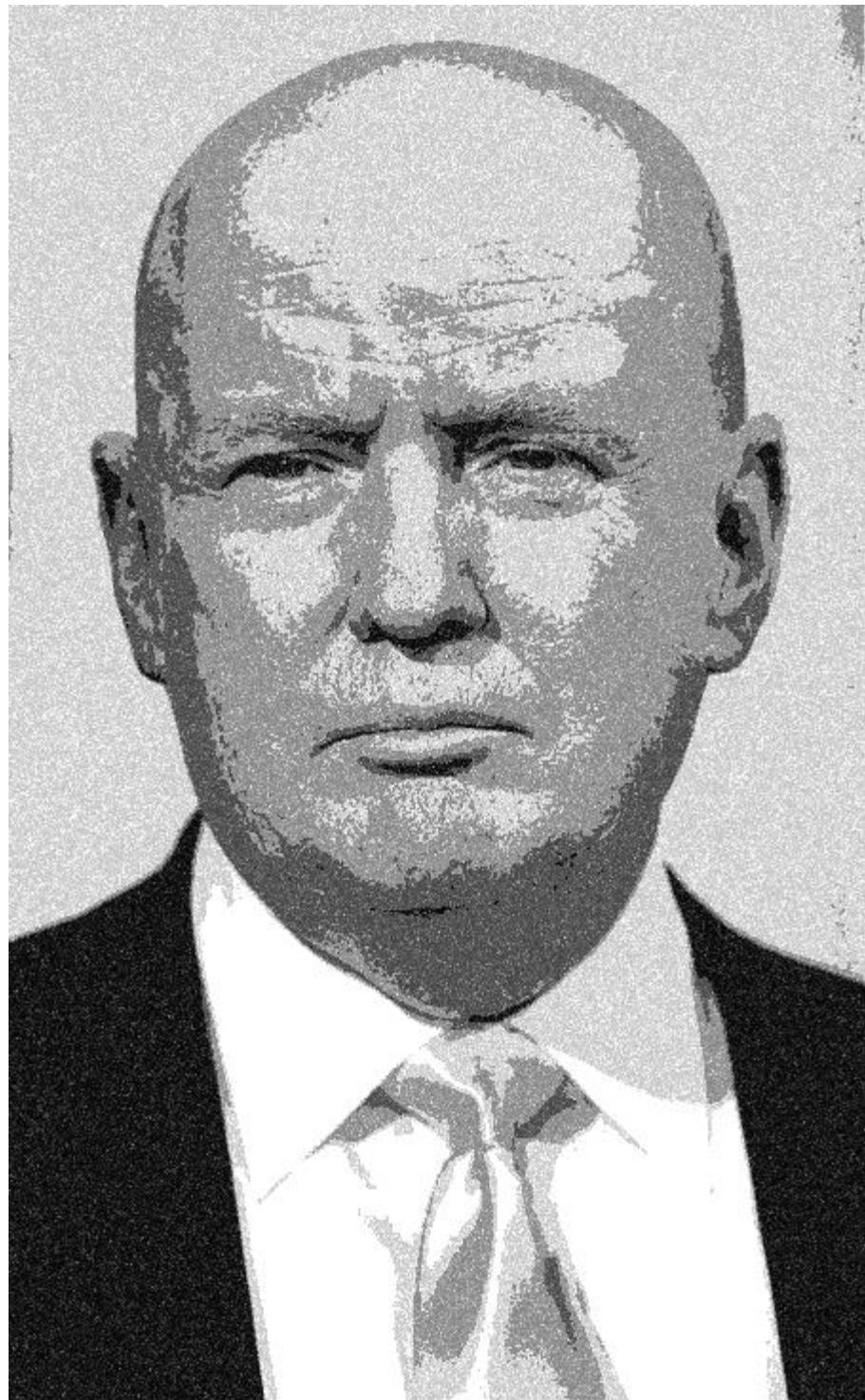
: (p2e\_hist\_a\_2\_b\_0.03.png) 0.03 b و a 2 هیستوگرام تصویر نویزی شده با



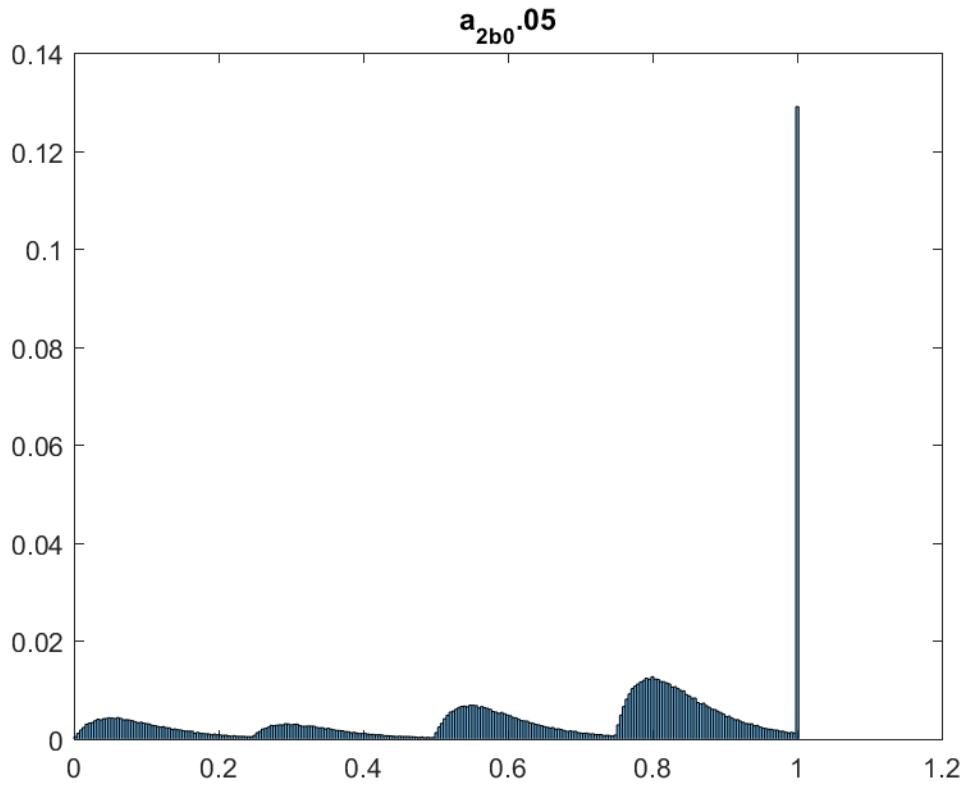
در اطراف ۵ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع Gamma ایجاد شده است.

در ابتدا اشاره کردیم که در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم. به همین دلیل در صفر و یک میزان هیستوگرام زیادتر است.

تصویر نویزی شده با  $a = 2$  و  $b = 0.05$ : (p2e\_a\_2\_b\_0.05.png)



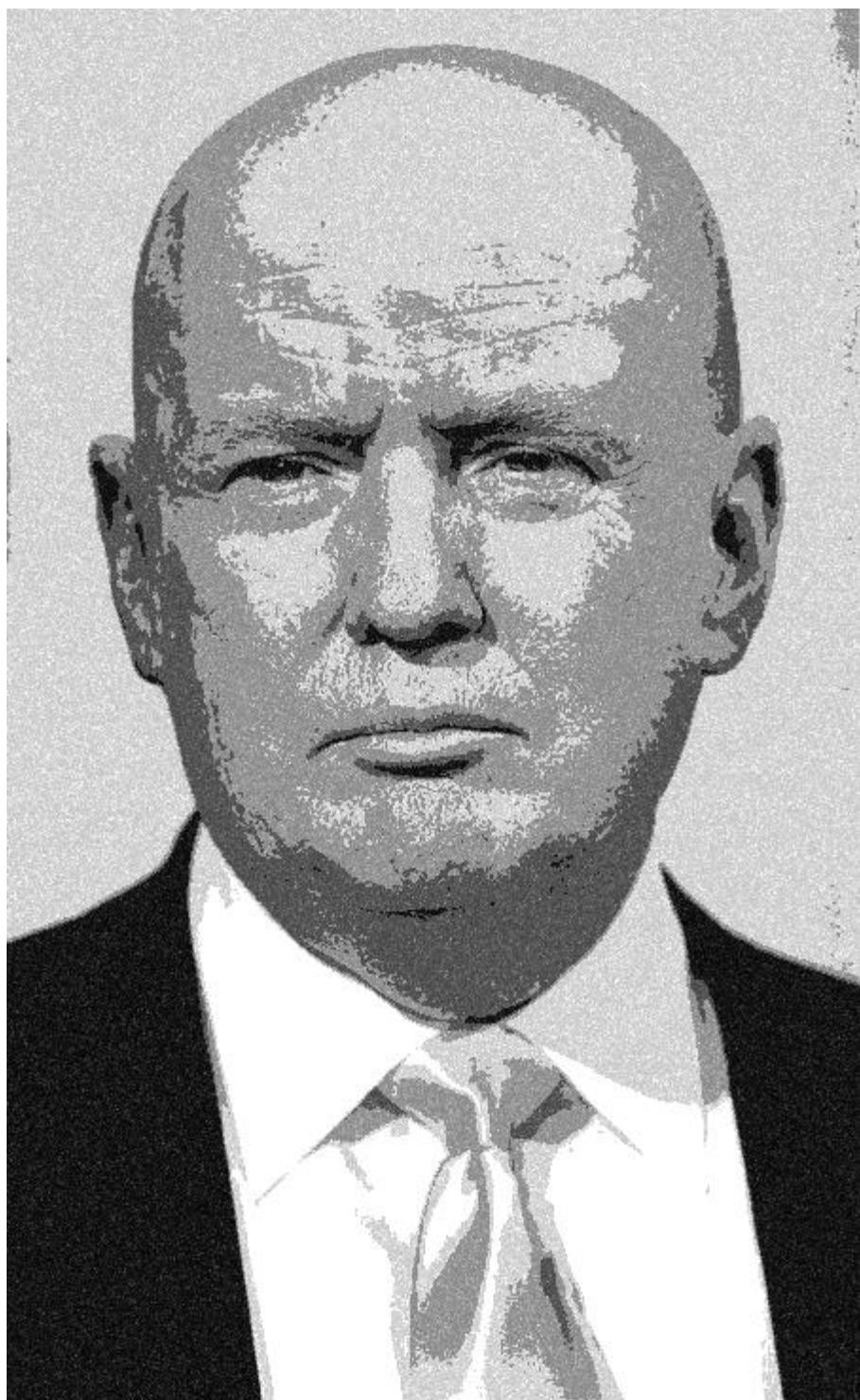
: (p2e\_hist\_a\_2\_b\_0.05.png) 0.05 b و a 2 هیستوگرام تصویر نویزی شده با



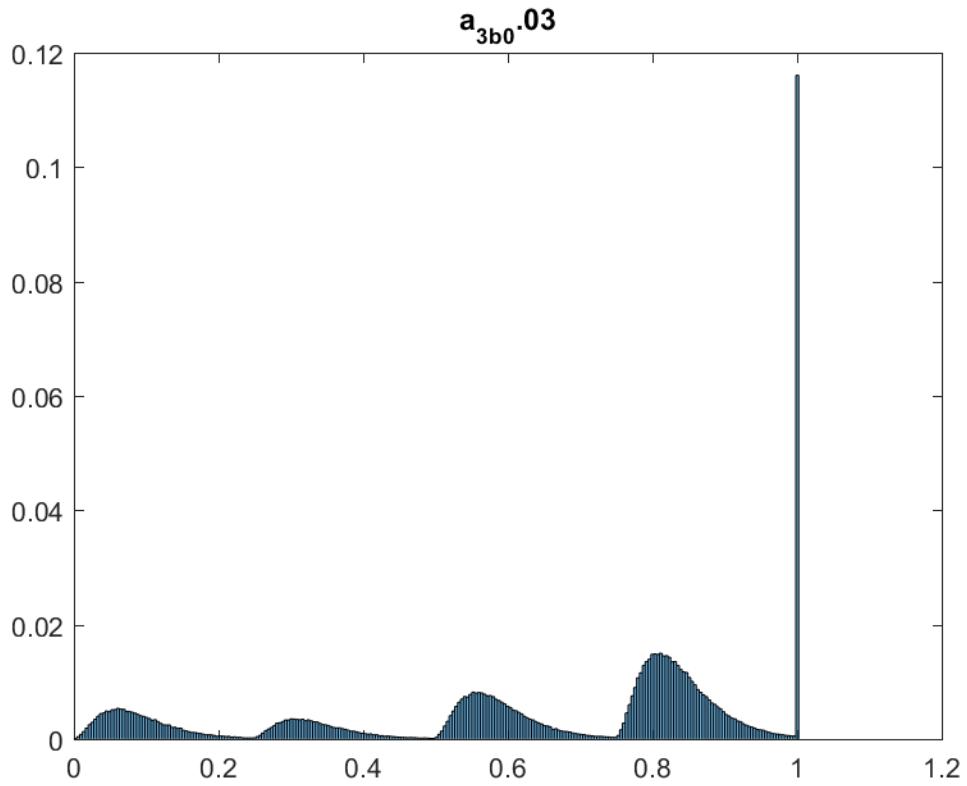
در اطراف ۵ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع Gamma ایجاد شده است.

در ابتدا اشاره کردیم که در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم. به همین دلیل در صفر و یک میزان هیستوگرام زیادتر است. با زیاد شدن  $a$  ارتفاع قله کمتر شده است و توزیع عریض‌تر شده است.

تصویر نویزی شده با  $a = 3$  و  $b = 0.03$ : (p2e\_a\_3\_b\_0.03.png)



: (p2e\_hist\_a\_3\_b\_0.03.png) 0.03 b 3 a با شده نویزی تصویر p2e\_hist\_a\_3\_b\_0.03.png



در اطراف ۵ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع Gamma ایجاد شده است.

در ابتدا اشاره کردیم که در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم. به همین دلیل در صفر و یک میزان هیستوگرام زیادتر است.

در بالا، بعد از هر تصویر نویزی توضیحاتی برای آن‌ها ارائه دادیم.

#### قسمت f

کدهای این قسمت در p2f.m و p2f\_func.m قرار دارد. در p2f\_func.m تابع زیر قرار دارد:

```

]function noisy_img = p2f_func(img, a)
]    % This function adds exponential noise to image.
% img is an image with pixels value in range [0,1]
% a,

```

این تابع یک تصویر با دیتاتایپ **double** می‌گیرد و با استفاده از نویزی که از توزیع Exponential استخراج می‌کند، تصویر را با توجه به پارامترهای ورودی نویزی می‌کند. در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم.

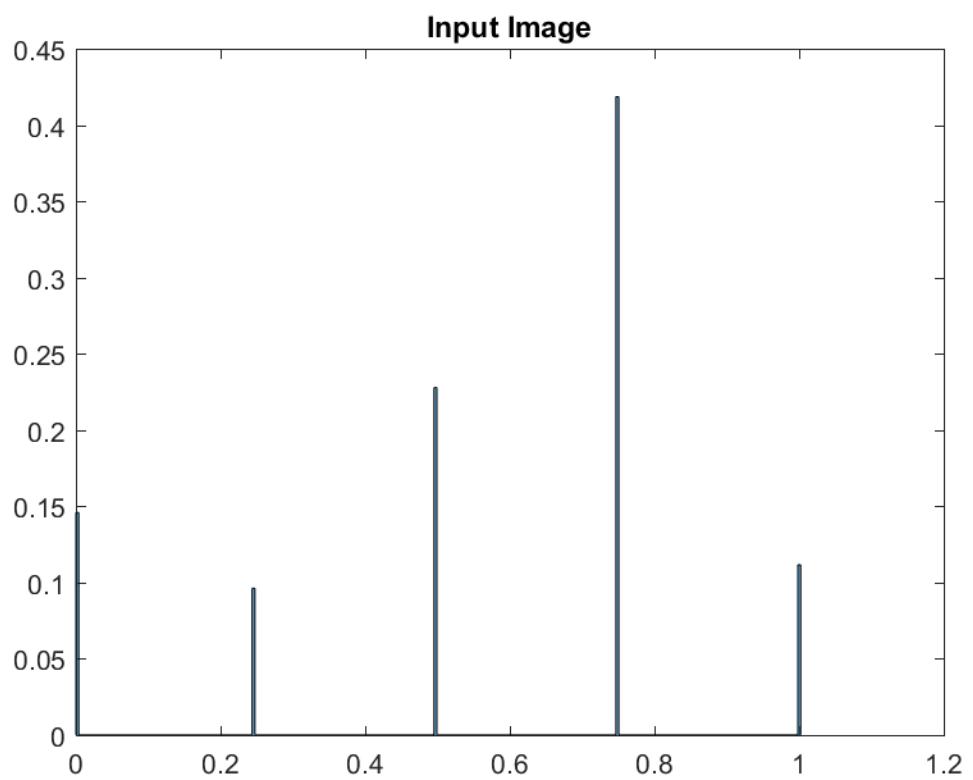
در p2f.m ، تصویر ورودی را از فایل می‌خوانیم سپس هیستوگرام آن را نمایش می‌دهیم و در ادامه با سه مقدار مختلف پارامترهای نویز، تصویر را نویزی می‌کنیم و تصویر نویزی شده و هیستوگرام آن را نمایش می‌دهیم.

در زیر خروجی‌های کد را مشاهده می‌کنید:

تصویر ورودی:



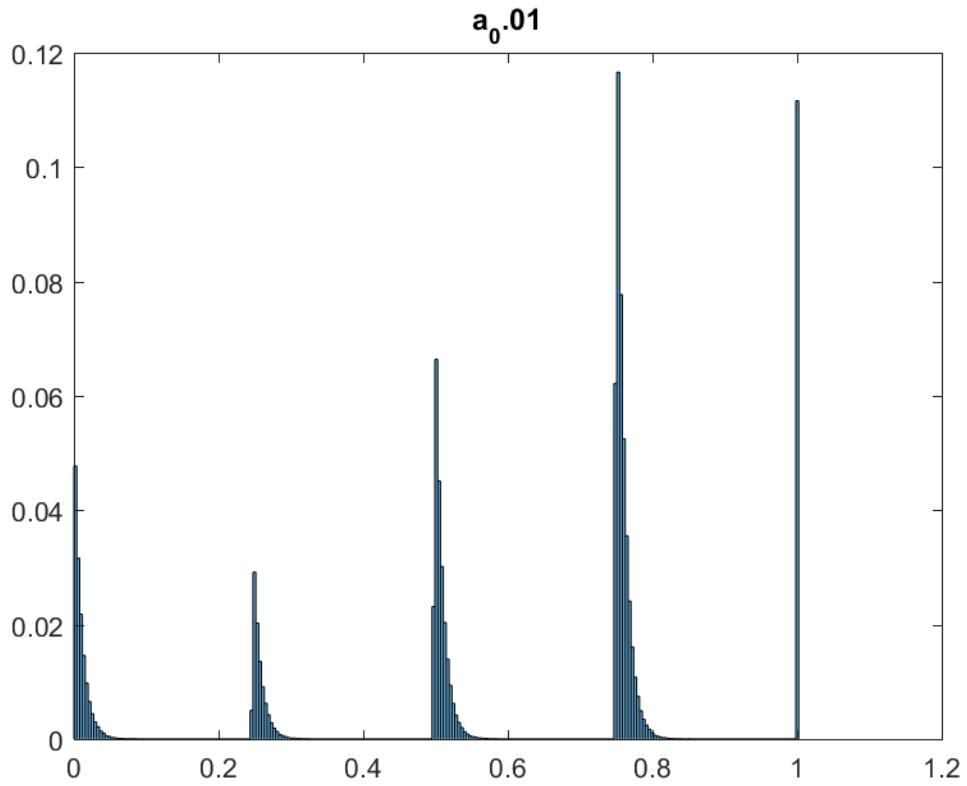
هیستوگرام تصویر ورودی:



تصویر نویزی شده با  $a = 0.01$ : (p2f\_a\_0.01.png)



: (p2f\_hist\_a\_0.01.png) 0.01 a 0.01 هیستوگرام تصویر نویزی شده با



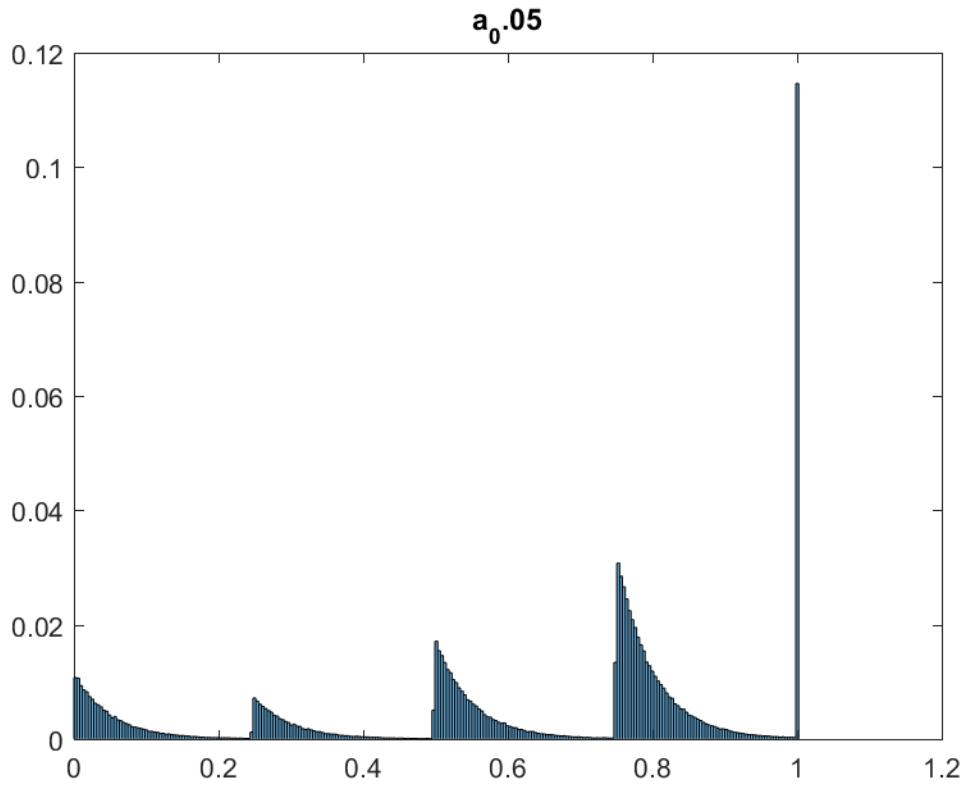
در اطراف ۵ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع Exponential ایجاد شده است.

در ابتدا اشاره کردیم که در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم. به همین دلیل در صفر و یک میزان هیستوگرام زیادتر است.

تصویر نویزی شده با  $a = 0.05$ : (p2f\_a\_0.05.png)



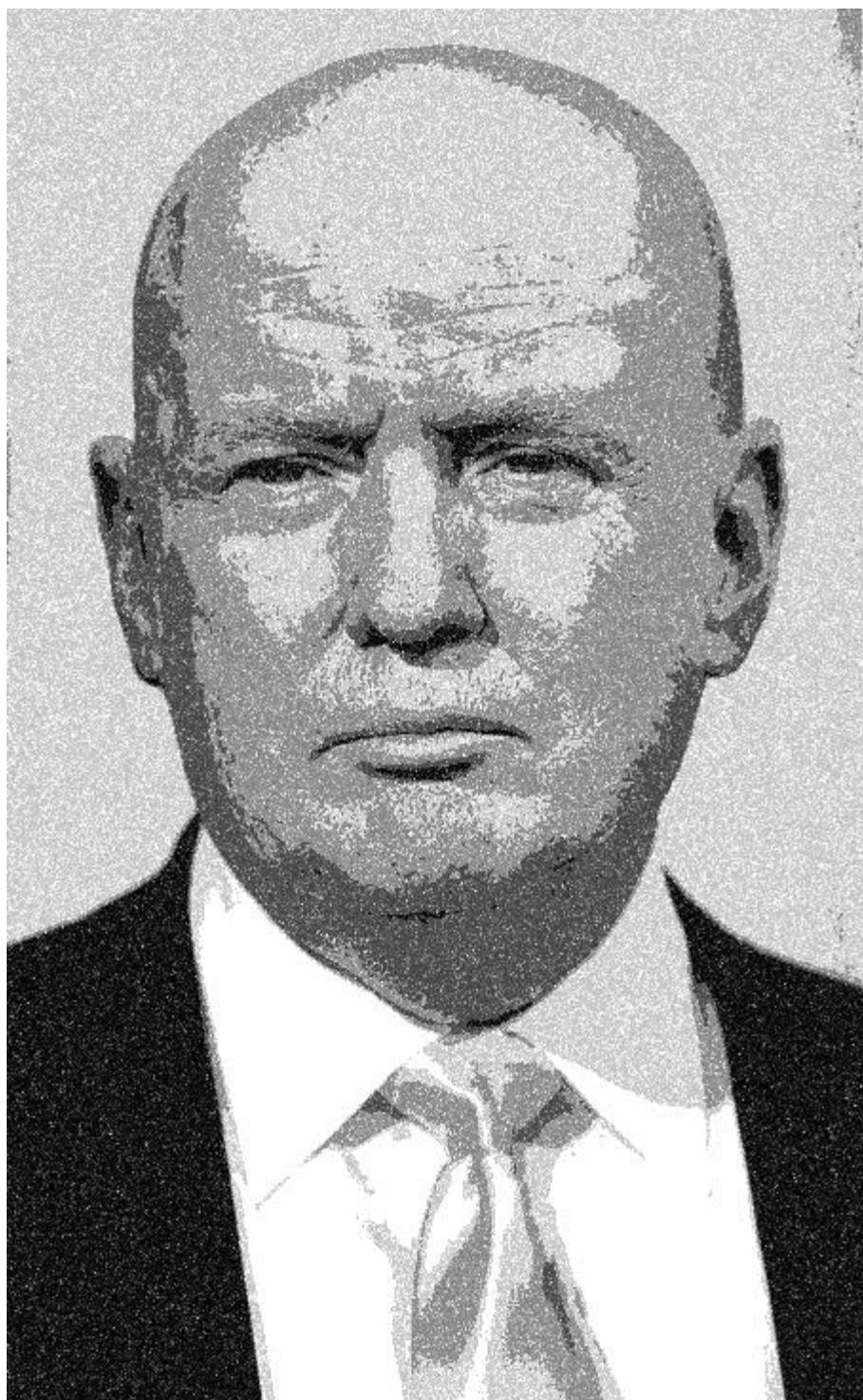
: (p2f\_hist\_a\_0.05.png) 0.05 a هیستوگرام تصویر نویزی شده با



در اطراف ۰ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع Exponential ایجاد شده است. میزان  $a$  بیشتر شده است و ارتفاع قله کاهش پیدا کرده است و توزیع پهن‌تر شده است و نویز بیشتری در تصویر دیده می‌شود.

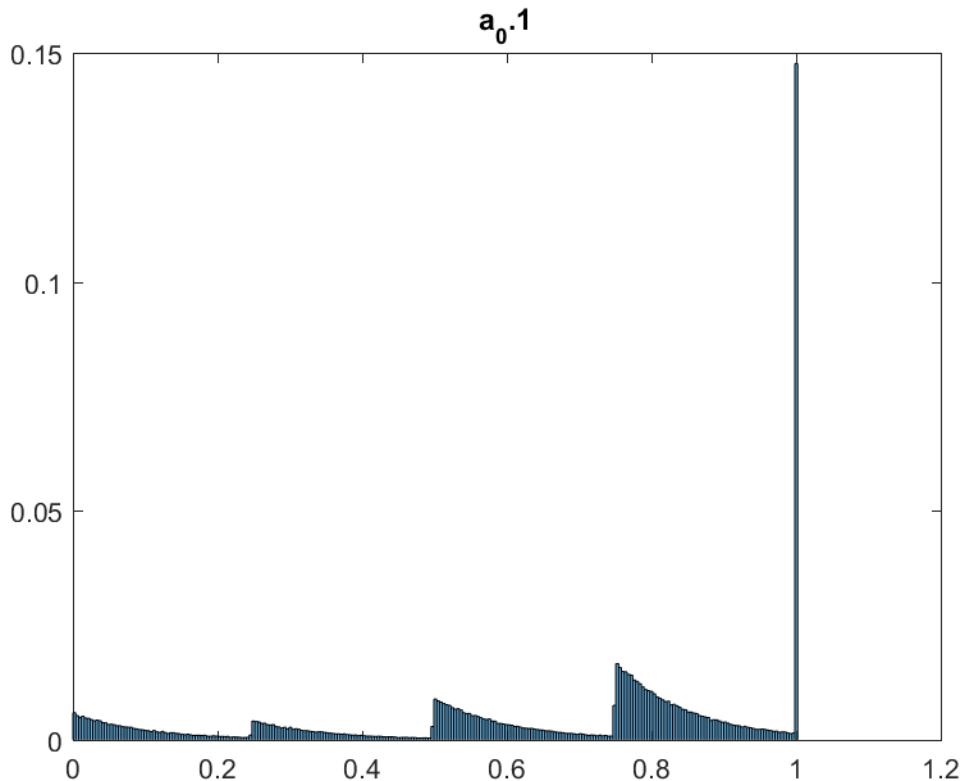
در ابتدا اشاره کردیم که در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم. به همین دلیل در صفر و یک میزان هیستوگرام زیادتر است.

: (p2f\_a\_0.1.png) ۰.۱ a تصویر نویزی شده با



: (p2f\_hist\_a\_0.1.png) 0.1 a هیستوگرام تصویر نویزی شده با

در بالا، بعد از هر تصویر نویزی توضیحاتی برای آن‌ها ارائه دادیم.



در اطراف ۵ شدت روشنایی در هیستوگرام تصویر بدون نویز، یک توزیع Exponential ایجاد شده است. میزان  $a$  بیشتر شده است و ارتفاع قله کاهش پیدا کرده است و توزیع پهن‌تر شده است و نویز بیشتری در تصویر دیده می‌شود.

در ابتدا اشاره کردیم که در صورتی که شدت روشنایی بیش از ۱ شود آن را برابر با یک قرار داده‌ایم و در صورتی که شدت روشنایی کمتر از ۰ شود آن را برابر با ۰ قرار می‌دهیم. به همین دلیل در صفر و یک میزان هیستوگرام زیادتر است.

### قسمت g

کدهای این قسمت در p2g\_func.m و p2g\_func.m قرار دارد. در p2g.m تابع زیر قرار دارد:

```

function img_out = p2g_func(img_in, k)
    % add turbulence effect to image
    % k: power of turbulence

```

این تابع یک تصویر را به عنوان ورودی می‌گیرد همچنین عدد  $k$  را دریافت می‌کند که میزان شدت Turbulence را مشخص می‌کند. ابتدا تصویر ورودی را ضمن انتقال مختصات به مرکز به حوزه‌ی فرکانس می‌برد و سپس حوزه‌ی فرکانس بر اساس تابع زیر ایجاد می‌کند:

$$H(u, v) = e^{-k(u^2 + v^2)^{5/6}}$$

و آن را در تبدیل فوریه ضرب درایه در درایه می‌کند و در انتها تبدیل معکوس فوریه ضمن انتقال مرکز به گوشه سمت چپ بالا را انجام می‌دهد.

در زیر خروجی‌های کد را مشاهده می‌کنید:

تصویر ورودی:



تصویر تخریب شده با  $k$  برابر با 0.0025 (Severe)



تصویر تخریب شده با  $k$  برابر با 0.001 (Mild)



تصویر تخریب شده با  $k$  برابر با 0.00025 (Low)



## h قسمت

کدهای این قسمت در p2h\_func.m و p2h.m قرار دارد. در p2h\_func.m تابع زیر قرار دارد:

```
function img_out = p2h_func(img_in, a, b, T)
    % add turbulence effect to image
    % a: motion in x direction
    % b: motion in y direction
    % T: amount of displacement in time
```

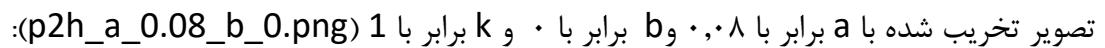
این تابع یک تصویر را به عنوان ورودی می‌گیرد همچنین عدهای  $a$  و  $b$  و  $T$  را دریافت می‌کند که میزان جا به جایی در راستای  $x$  و  $y$  و زمانی که جایه‌جایی زمان برده است تا انجام شود را مشخص می‌کند. ابتدا تصویر ورودی را ضمن انتقال مختصات به مرکز به حوزه‌ی فرکانس می‌برد و سپس degradation را حوزه‌ی فرکانس بر اساس تابع زیر ایجاد می‌کند:

$$H(u, v) = \frac{T}{\pi(ua + vb)} \sin[\pi(ua + vb)] e^{-j\pi(ua + vb)}$$

و آن را در تبدیل فوریه ضرب درایه در درایه می‌کند و در انتهای تبدیل معکوس فوریه ضمن انتقال مرکز به گوشه سمت چپ بالا را انجام می‌دهد.

در  $p2h.m$  به ازای مقدارهای مختلف  $a, b, T$  تابع بالا فرآخوانده می‌شود. در زیر خروجی‌های کد را مشاهده می‌کنید:

تصویر ورودی:

: تصویر تخریب شده با  $a$  برابر با  $0.08$  و  $b$  برابر با  $0$  و  $k$  برابر با  $1$



تصویر تخریب شده با  $a$  برابر با  $0.08$  و  $b$  برابر با  $1$  و  $k$  برابر با  $0$  (p2h\_a\_0\_b\_0.08.png)



تصویر تخریب شده با  $a$  برابر با  $0,08$  و  $b$  برابر با  $0,08$  و  $k$  برابر با  $1$ : (p2h\_a\_0.08\_b\_0.08.png)



### تمرین ۳

کدهای این قسمت در p3 قرار دارد.

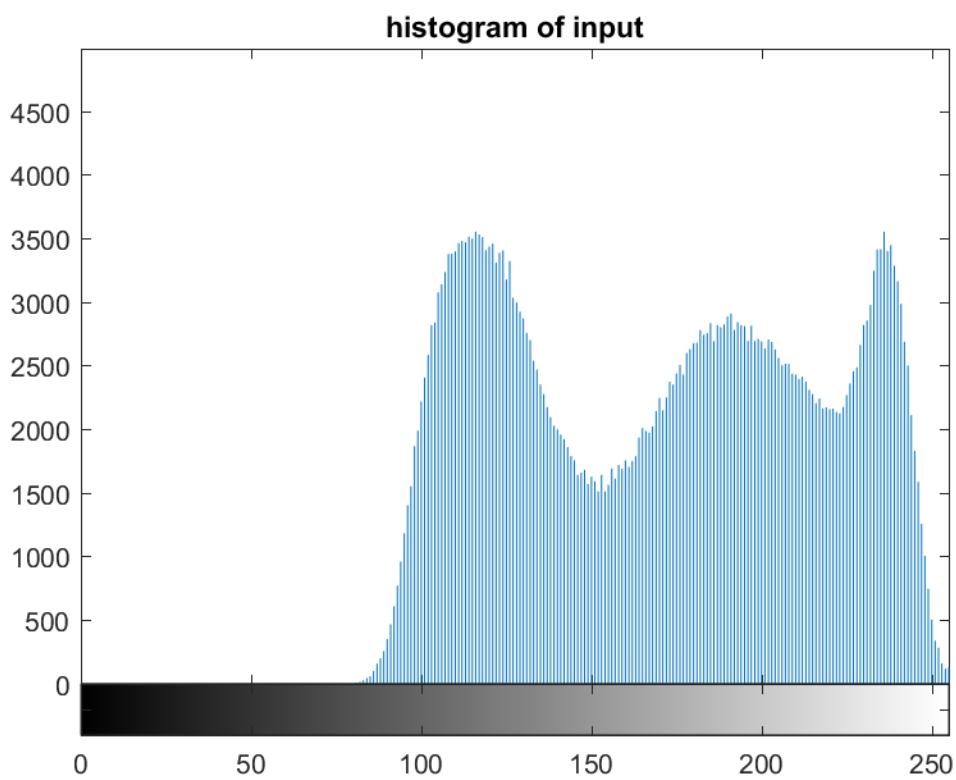
#### قسمت a

کد این قسمت در p3a.m قرار دارد. ابتدا تصویر ورودی و هیستوگرام آن را رسم می کنم، سپس یک ناحیه‌ی یک نواخت از تصویر را جهت بررسی نوع و پارامترهای نویز انتخاب می‌کنم و هیستوگرامش را رسم می‌کنم:

تصویر ورودی:



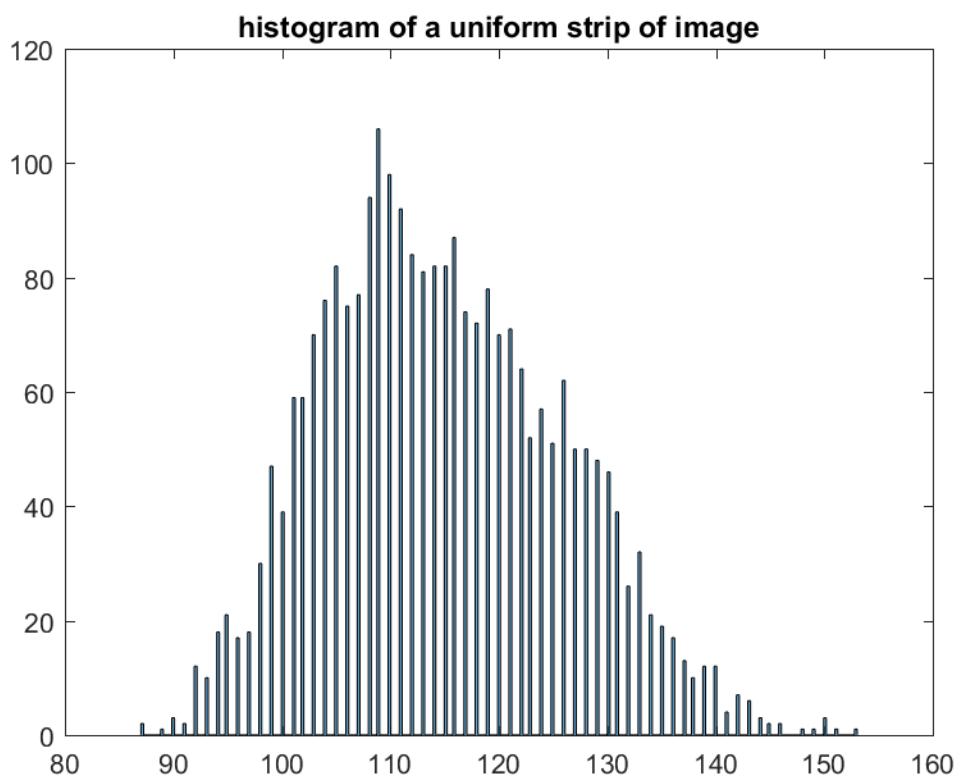
هیستوگرام تصویر ورودی:



یک قسمت یکنواخت از تصویر:



هیستوگرام قسمت یکنواخت از تصویر:



همین طور که در هیستوگرام بالا مشاهده می‌شود، این نویز بسیار به نویز گاووسی نزدیک است، به همین دلیل از یک فیلتر گاووسی برای رفع نویز استفاده می‌کنم. بر اساس هیستوگرام بالا و آزمون خطا واریانس برابر با ۱.۱ را انتخاب می‌کنم.

در تصویر زیر تصویر فیلتر شده را مشاهده می‌کنید:



ممکن از فرض شود در تصویر فیلتر شده‌ی بالا کمی نویز وجود دارد، در حالی که اینگونه نیست، زیرا تصویر اصلی را گوگل کردم و شبیه تصویر حاصل از فیلتر بالا است.

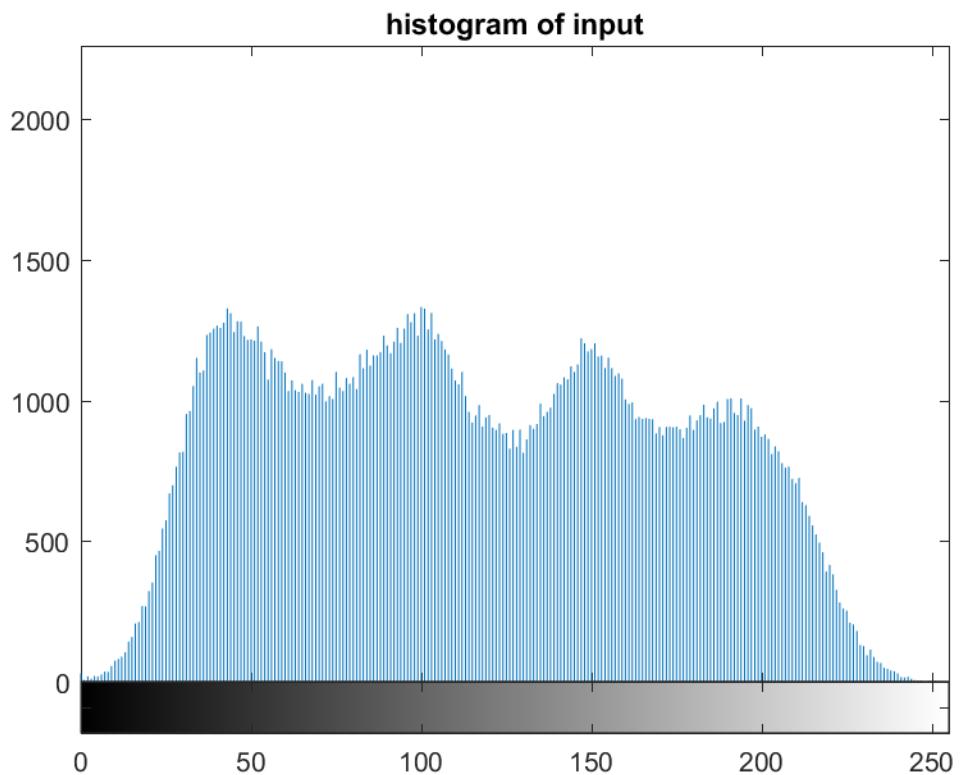
### ب) قسمت

کد این قسمت در p3b.m قرار دارد. نویز از نوع salt&pepper بر خلاف بسیاری از نویزهای addictive را حتی توسط چشم قابل شناسایی است و نیازی به تحلیل‌های آماری ندارد. در کد این قسمت تصویر و هیستограм آن را رسم کرده‌ام، همچنین یک ناحیه‌ی یکنواخت را انتخاب کرده‌ام و رسم کرده‌ام. در ادامه یک فیلتر میانه با اندازه‌ی پنجرهای  $3^*3$  اعمال کرده‌ام. در زیر خروجی‌های کد را مشاهده می‌کنید:

تصویر ورودی:



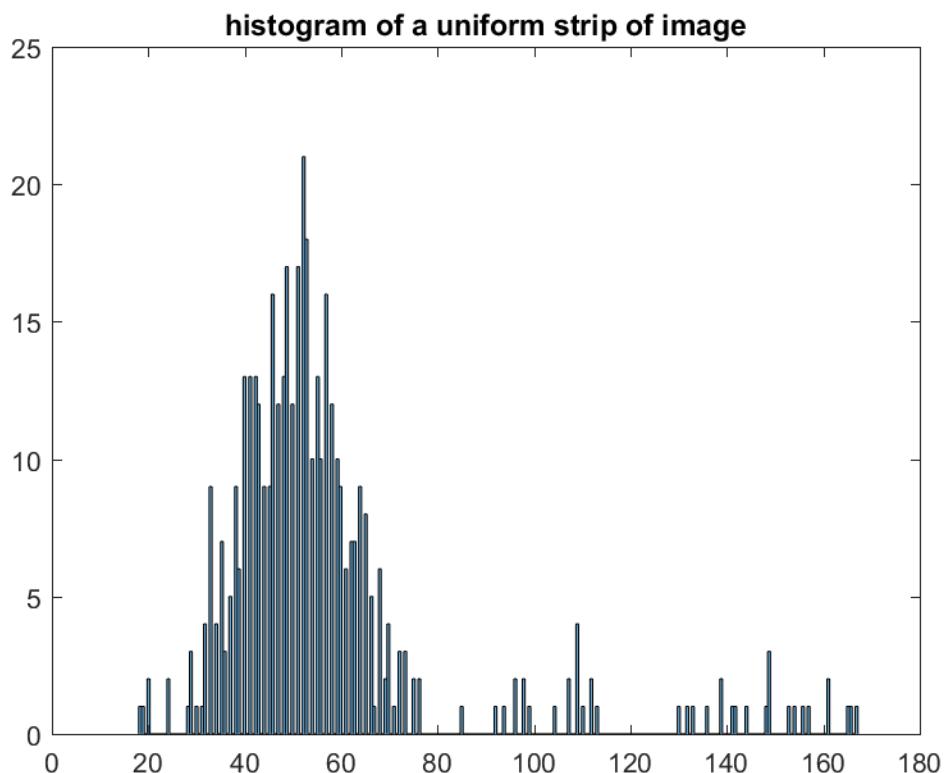
هیستوگرام تصویر ورودی:



یک قسمت یکنواخت از تصویر:



هیستوگرام قسمت یکنواخت از تصویر:



همین طور که در هیستوگرام ناحیه یکنواخت تیره رنگ مشاهده می شود، شدت روشنایی هایی با مقدار زیاد و یا بسیار کم خارج از محدوده ای شدت روشنایی های عمدی اصلی ناحیه یکنواخت دیده می شوند. که نشان دهنده نویزهای Salt&Pepper است.

در تصویر زیر تصویر فیلتر شده را مشاهده می کنید که پارامترهای آن را با آزمون و خطأ و بررسی تصویر پیدا کردم (p3b\_filtered\_img.png):



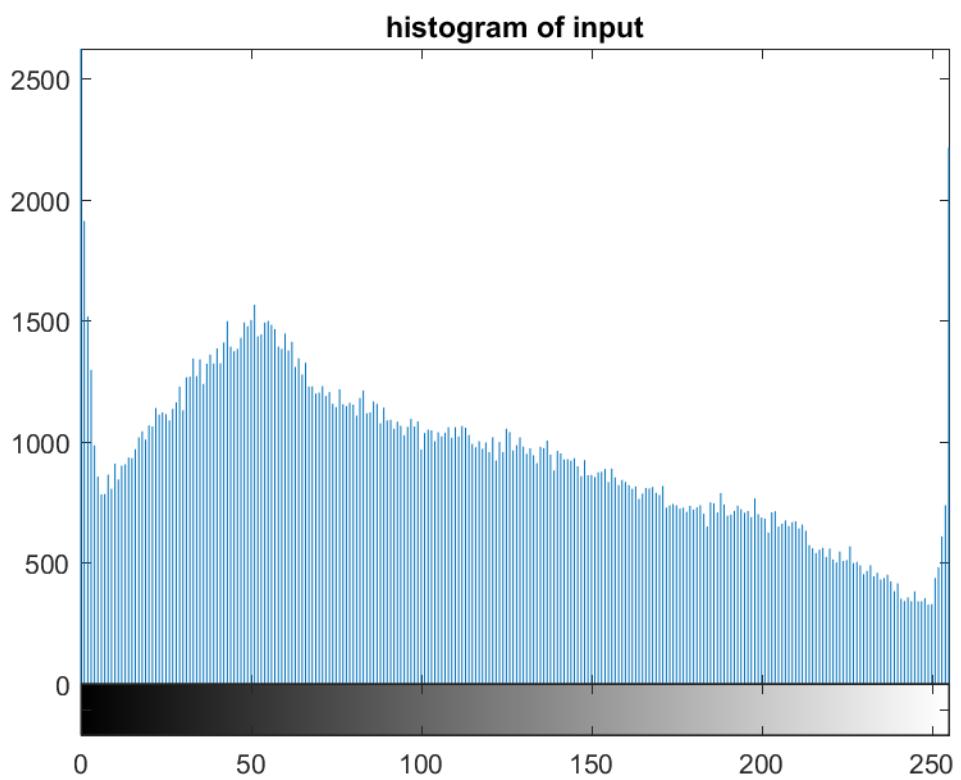
### قسمت ۳

کد این قسمت در p3c.m قرار دارد. ابتدا تصویر ورودی و هیستوگرام آن را رسم می کنم، سپس یک ناحیه‌ی یک نواخت از تصویر را جهت بررسی نوع و پارامترهای نویز انتخاب می کنم و هیستوگرامش را رسم می کنم:

تصویر ورودی:



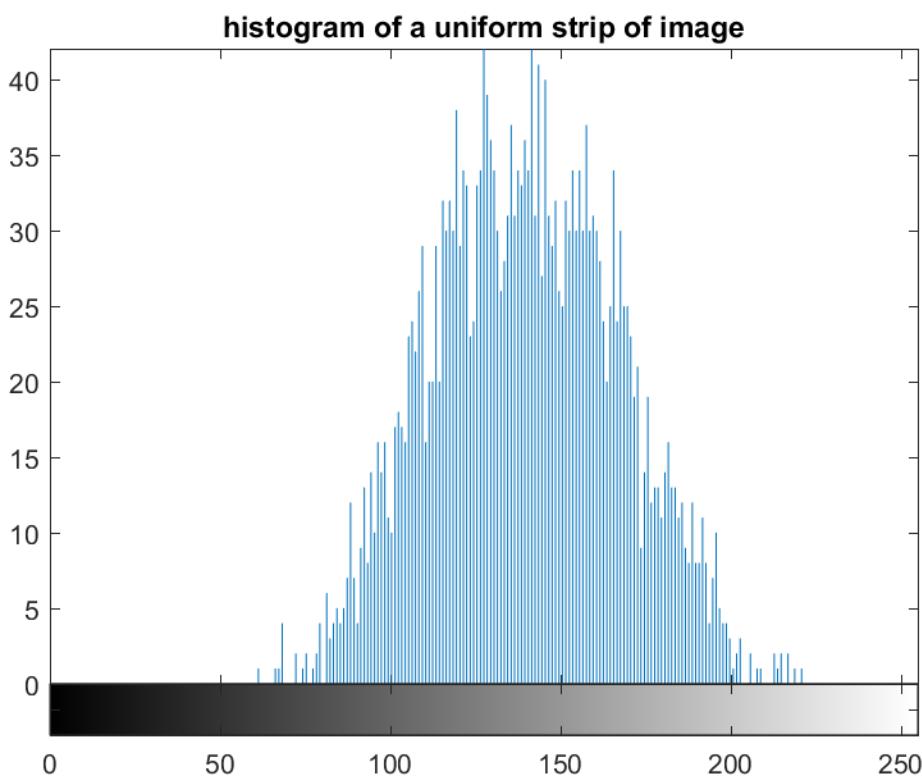
هیستوگرام تصویر ورودی:



یک قسمت یکنواخت از تصویر:



هیستوگرام قسمت یکنواخت از تصویر:



همین طور که در هیستوگرام ناحیه‌ی یکنواخت استخراج شده از تصویر مشاهده می‌شود، این نویز بسیار به نویز گاووسی نزدیک است، همچنین در هیستوگرام تصویر ورودی در انتهای و ابتدای هیستوگرام دو پیک بزرگ وجود دارد که ممکن است حاصل فلفل و نمک باشد، به همین دلیل از **یک فیلتر گاووسی و میانه** به طور همزمان برای رفع نویز استفاده می‌کنم. بر اساس هیستوگرام بالا و آرمون خطای واریانس برابر با  $8,0$  و پنچرهی میانه را برابر  $3 \times 3$  انتخاب می‌کنم.

در تصویر زیر تصویر فیلتر شده را مشاهده می‌کنید:



#### d قسمت

کد این قسمت در p3d.m قرار دارد. در این تصویر دو نویز دیده می‌شود: نویز سینوسی و موج addictive که باید با استفاده از روش‌های آماری نوعش را پیدا کنیم.

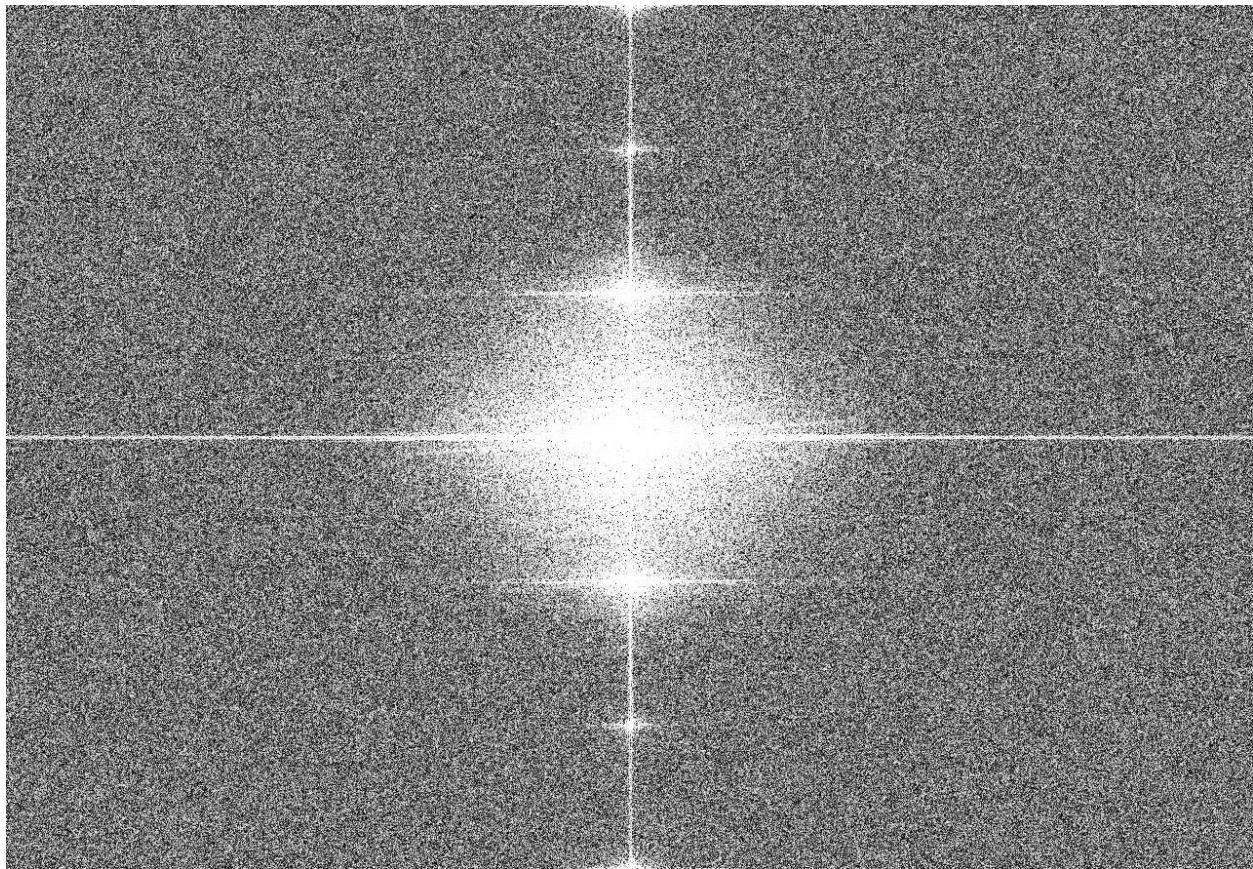
در کد ابتدا تصویر را می‌خوانم، ابتدا موج سینوسی را حذف می‌کنیم. از آنجایی که خطوطی افقی در تصویر وجود دارد، در حوزه فرکانس با یک خط عمودی با amplitude بالا مواجه هستیم. به همین دلیل یک فیلتر عمودی ناج (بر اساس کتاب ناج فقط دایره نیست و می‌تواند مربع و مستطیل و ... هم باشد) که شامل مرکز مختصات نمی‌شود طراحی می‌کنیم و آن را در حوزه‌ی فرکانس در فوریه‌ی تصویر ضرب می‌کنیم:

در زیر تصویر ورودی، phase spectrum و spectrum آن و فیلتر طراحی شده و تصویر حاصل از فیلتر را مشاهده می‌کنید:

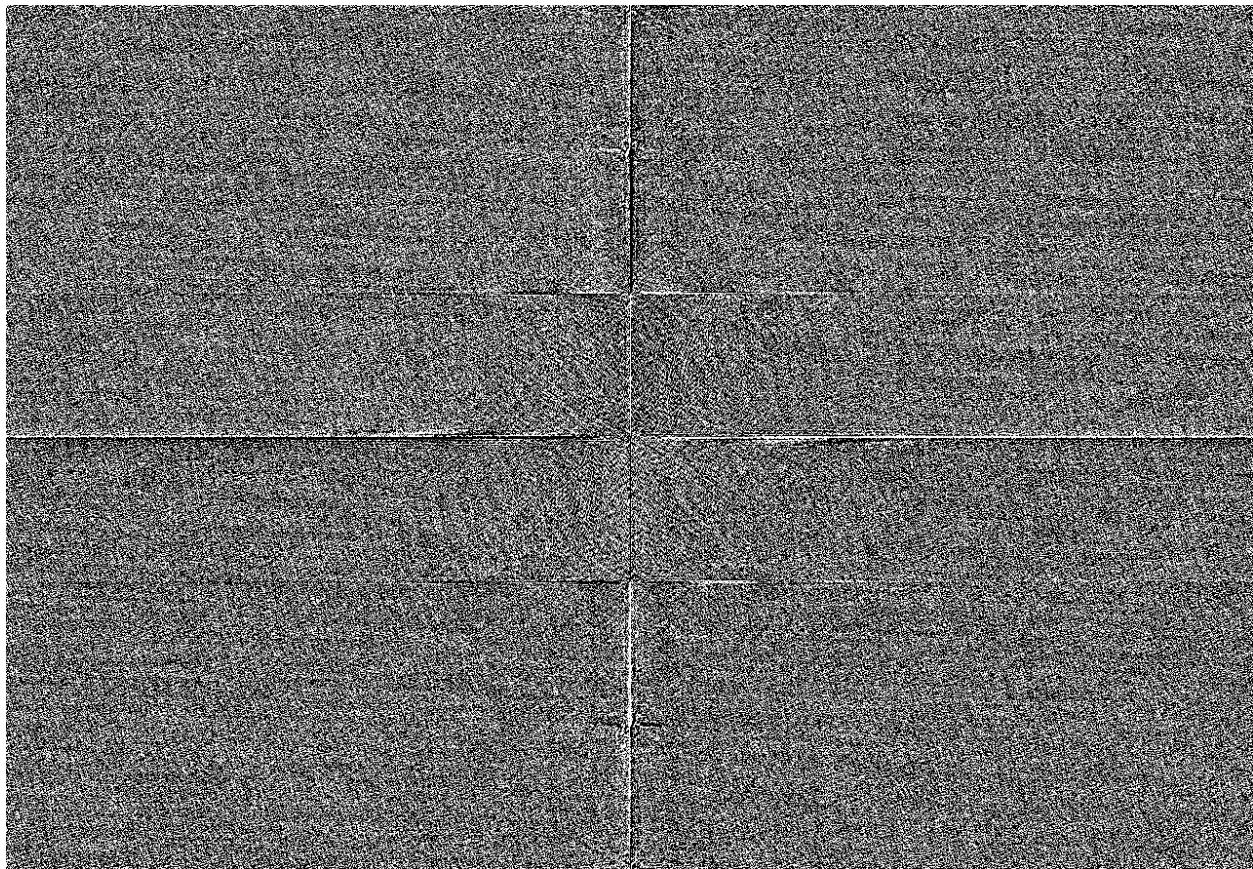
تصویر ورودی:



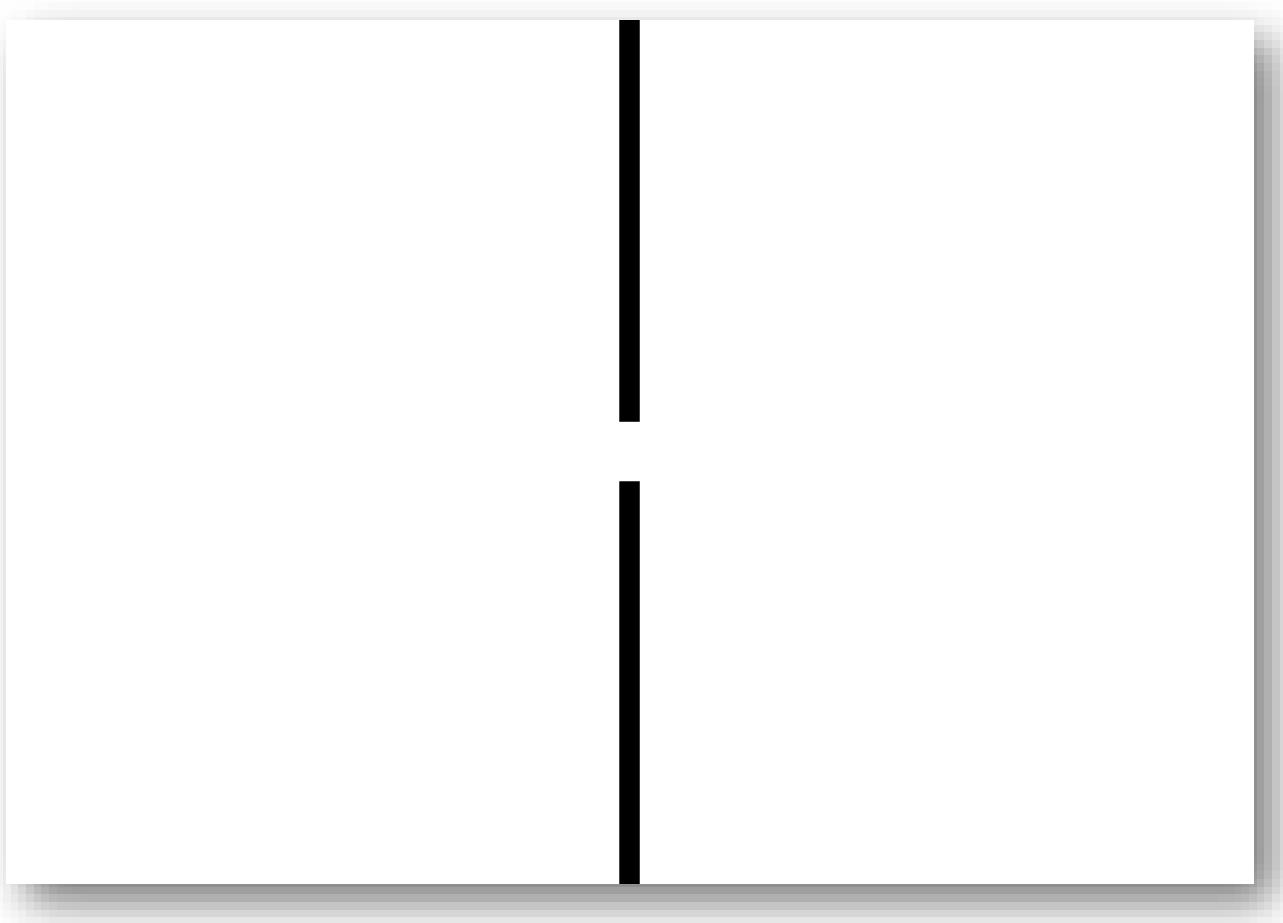
: (p3d\_spectrum.png) تصویر ورودی Spectrum



: (p3d\_phase.png تصویر ورودی Phase



فیلتر طراحی شده (p3d\_filter\_1.png)



تصویر حاصل از اعمال فیلتر که منجر به حذف خطهای عمودی شده  
است:(p3d\_filtered\_image\_1.png)

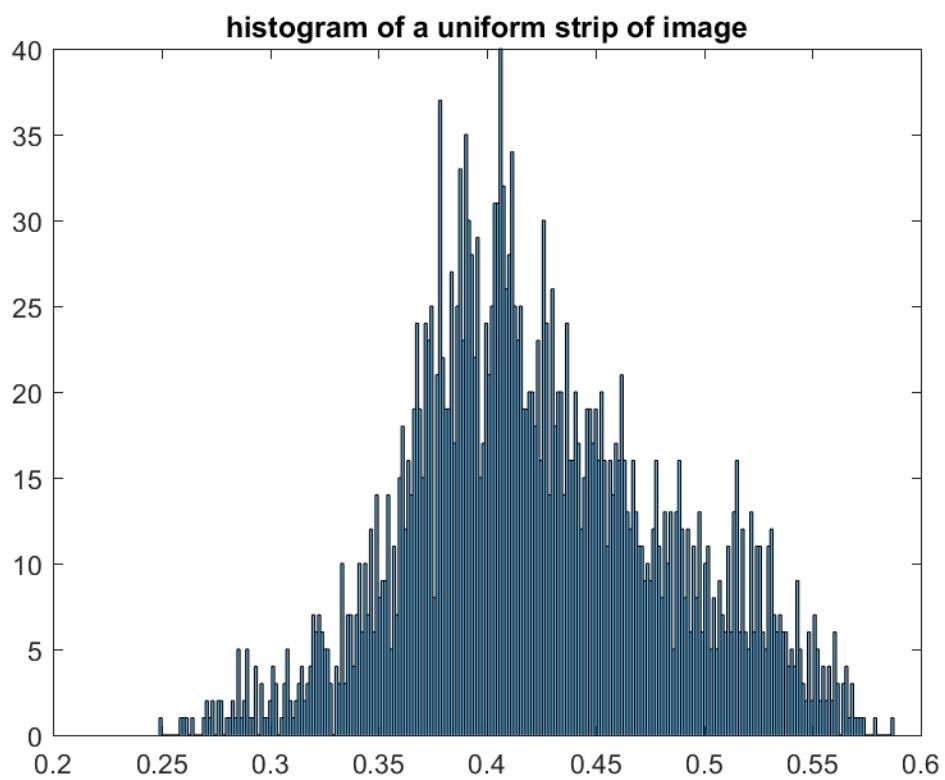


تصویر نویز دیگری هم دارد که برای تشخیص آن یک ناحیهٔ یکنواخت را انتخاب می‌کنیم و هیستوگرام آن را  
رسم می‌کنیم:

ناحیهٔ یکنواخت (p3d\_strip.png)



هیستوگرام ناحیهٔ یکنواخت (p3d\_hist\_strip\_image.png)



همین طور که مشاهده می شود این ناحیه هیستوگرامی شبیه به توزیع گووسی و Rayleigh دارد. به همین خاطر از یک فیلتر گاوسی برای حذف نویز استفاده می کنیم. بر اساس نمودار بالا و آزمون و خطای میزان ۱,۲ را برای انتخاب می کنیم.

در زیر خروجی نهایی را بعد از اعمال دو فیلتر بر روی دو نویز مشاهده می‌کنید (p3d\_filtered\_img\_final.png).



تصویر ورودی



تصویر خروجی



## تمرین ۴

کدها و خروجی‌های این سوال در p4 قرار دارد.

### قسمت a

کد این قسمت در p4a.m قرار دارد، ابتدا تصویر را از فایل می‌خوانیم سپس هر ستون را کپی می‌کنیم و کپی ستون را در سمت راست ستون قرار می‌دهیم. در ادامه هر سطر، تصویر جدید را کپی می‌کنیم و کپی را در پایین سطر قرار می‌دهیم. به این ترتیب، سایز تصویر را دو برابر می‌کنیم. در زیر خروجی تصویر را مشاهده می‌کنید (برای دیدن تصویرها در اندازه‌ی واقعی به p4 مراجعه کنید):

وروودی:



: (p4a.png) خروجی



#### قسمت b

کد این قسمت در p4b.m قرار دارد، ابتدا تصویر را از فایل می‌خوانیم سپس بین هر دو ستون میانگین دو ستون را قرار می‌دهیم. در ادامه بین هر دو سطر تصویر جدید، میانگین دو سطر را وارد می‌کنیم. به این ترتیب، سایز تصویر را دو برابر می‌کنیم. در زیر خروجی تصویر را مشاهده می‌کنید (برای دیدن تصویرها در اندازه‌ی واقعی به p4 مراجعه کنید):

وروودی:



خروجی (p4b.png)



### قسمت C

کد این قسمت در p4c.m قرار دارد، ابتدا تصویر را از فایل می‌خوانیم ابتدا interpolation انجام می‌دهم و از میان همسایگی پیکسل، شدت روشنایی می‌کنم سپس برای نقاط جدید nearest neighbor تصویر را پیدا

پیکسلی را انتخاب می کنم که مقدارش به مقدار interpolation نزدیکتر است. در زیر خروجی تصویر را مشاهده می کنید (برای دیدن تصویرها در اندازه‌ی واقعی به p4 مراجعه کنید):

وروودی:



: (p4c.png) خروجی



## تمرین ۵

### قسمت a

کد این قسمت در p5a.m قرار دارد. در این کد از SWIRL برای Warping استفاده می‌کنیم. در زیر رابطه‌های آن را مشاهده می‌کنید:

SWIRL

$$\begin{aligned}x(u, v) &= (u - x_0) \cos(\theta) + (v - y_0) \sin(\theta) + x_0; \\y(u, v) &= -(u - x_0) \sin(\theta) + (v - y_0) \cos(\theta) + y_0; \\r &= ((u - x_0)^2 + (v - y_0)^2)^{1/2}, \theta = \pi r / 512.\end{aligned}$$

تصویر ورودی:



تصویر خروجی:



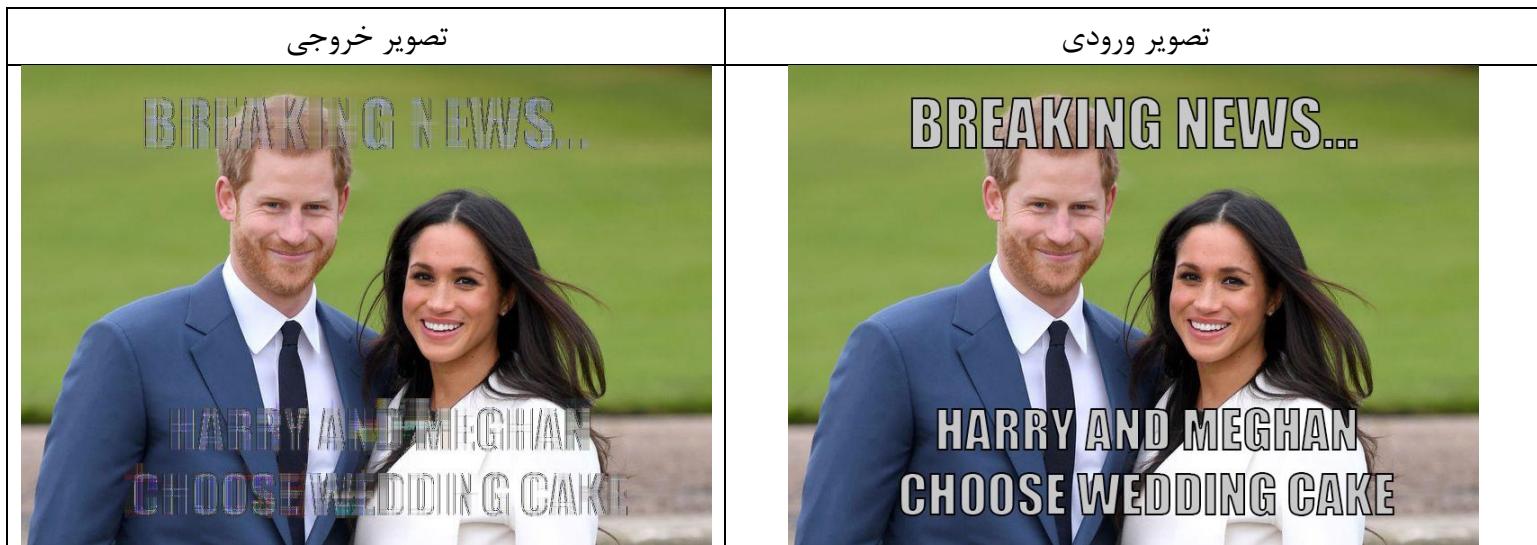
## قسمت b

کد این قسمت در 5b.m قرار دارد. در کد، ابتدا تصویر را می‌خوانم، سپس ناحیه‌هایی که در آن تاشدگی قرار دارد را محدود می‌کنم به یک سری مستطیل که تاشدگی‌ها درون آن قرار دارند. سپس بر اساس شدت روشنایی در هر مستطیل مشخص می‌کنم یک نقطه در مستطیل جز تاشدگی است یا خیر. در نقاط تاشدگی از چهار طرف *linear interpolation* شدت روشنایی نقطه‌ی تاشدگی را تخمین می‌زنم. برای دیدن تصویر در اندازه‌ی اصلی به p5 مراجعه کنید.



### قسمت c

کد این قسمت در p5c.m قرار دارد. در این قسمت مانند قسمت قبل عمل می‌کنیم. به همین دلیل روش انجام کار را دوباره توضیح نمی‌دهم. برای دیدن خروجی در اندازه‌ی اصلی به p5 مراجعه کنید:



### قسمت d

کد این قسمت در p5d.m قرار دارد. از آنجا که به روش خاصی برای انجام این سوال اشاره نشده است از این روش استفاده می‌کنم: ابتدا با تصویر ترامپ شروع می‌کنم و ۱۰۰ بار این کار را انجام می‌دهیم. هر بار ۱۵۰۰۰ پیکسل به صورت تصادفی از تصویر محمود انتخاب می‌کنم و در تصویر میانی جایگزین می‌کنم. در هر ۱۰ مرحله تصویر را رسم می‌کنم:



## تمرین ۶

کدهای این قسمت در پوشه‌ی p6 قرار دارد.

### قسمت a

کد این قسمت در p6a.m قرار دارد. در این قسمت ابتدا تصویر ورودی را می‌خوانیم. سپس آن را به دیتابیپ double تبدیل می‌کنیم. سپس با استفاده از تابع fft2 و fftshift ضمن انتقال مرکز مختصات به مرکز صفحه تبدیل فوریه‌ی تصویر را می‌گیریم. سپس spectrum و phase angle را ترسیم می‌کنیم.

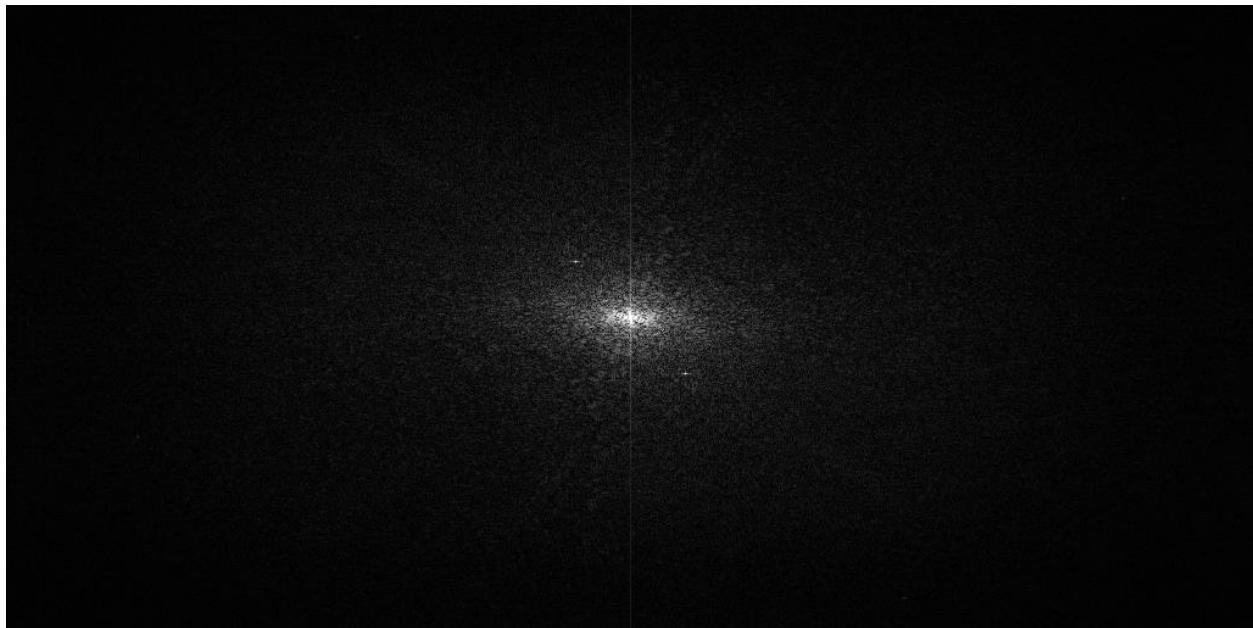
در spectrum یک جفت نقطه‌ی peak دیده می‌شود که با استفاده از Notch فیلترهای دایره‌ای فیلتری برای برطرف کردن آن‌ها ایجاد می‌کنیم.

در ادامه با ضرب فیلتر در تبدیل فوریه و تبدیل فوریه‌ی معکوس تصویر فیلتر شده را در حوزه‌ی spatial ایجاد می‌کنیم. در زیر خروجی‌های کد را مشاهده می‌کنید:

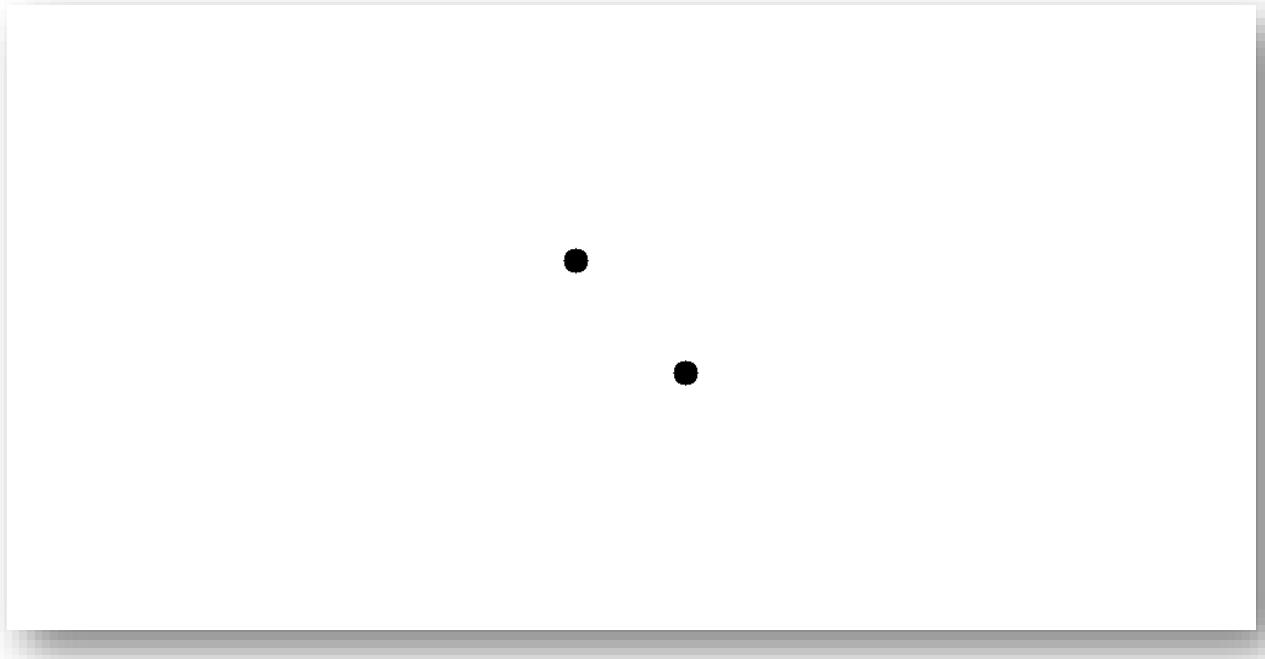
تصویر ورودی:



: (p6a\_spectrum.png) Spectrum تصویر ورودی



فیلتر ایجاد شده (p6a\_filter.png)



تصویر حاصل از اعمال فیلتر (p6a\_filtered\_image.png)



#### قسمت b

کد این قسمت در p6b.m قرار دارد. در این قسمت ابتدا تصویر ورودی را می‌خوانیم. سپس آن را به دیتابایپ double تبدیل می‌کنیم. سپس با استفاده از تابع fftshift و fft2 ضمن انتقال مرکز مختصات به مرکز صفحه تبدیل فوریه‌ی تصویر را می‌گیریم. سپس spectrum و phase angle را ترسیم می‌کنیم.

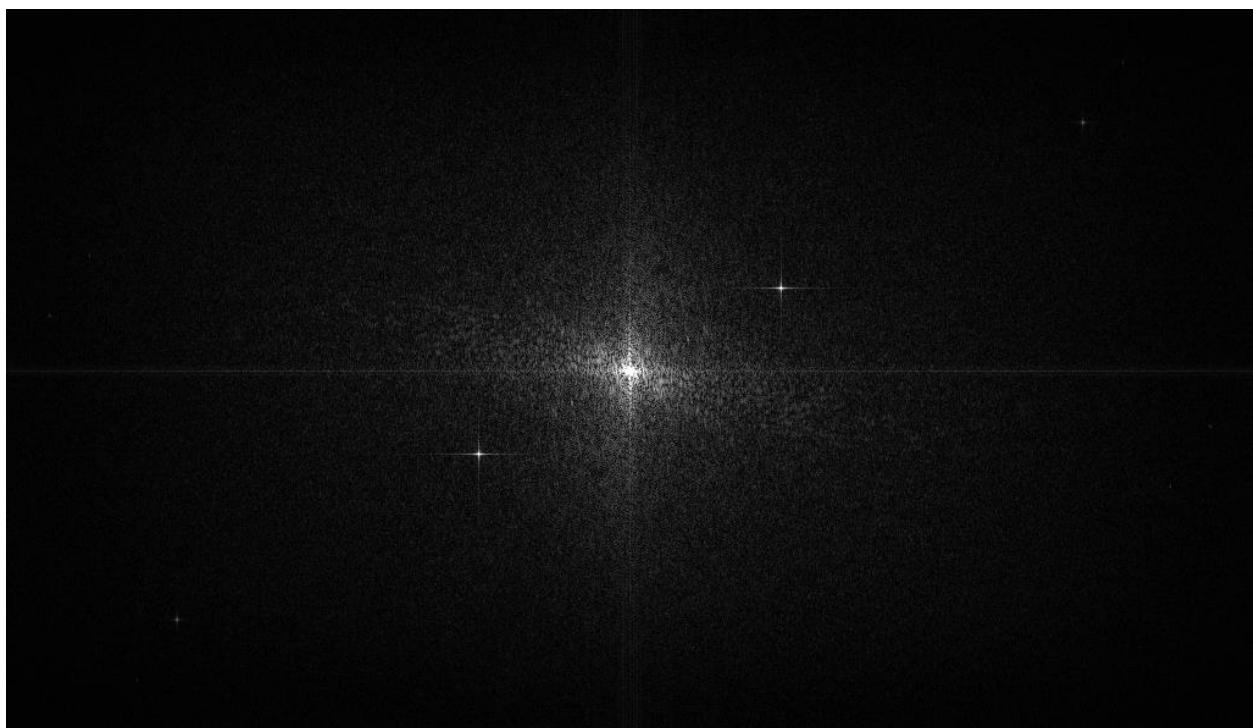
در این قسمت دو جفت نقطه‌ی peak دیده می‌شود که با استفاده از Notch فیلترهای دایره‌ای فیلتری برای برطرف کردن آن‌ها ایجاد می‌کنیم. (ناج فیلتر می‌توان هر شکلی باشد، مربع، مستطیل، دایره، بر اساس گفته‌ی کتاب)

در ادامه با ضرب فیلتر در تبدیل فوریه و تبدیل فوریه‌ی معکوس تصویر فیلتر شده را در حوزه‌ی spatial ایجاد می‌کنیم. در زیر خروجی‌های کد را مشاهده می‌کنید:

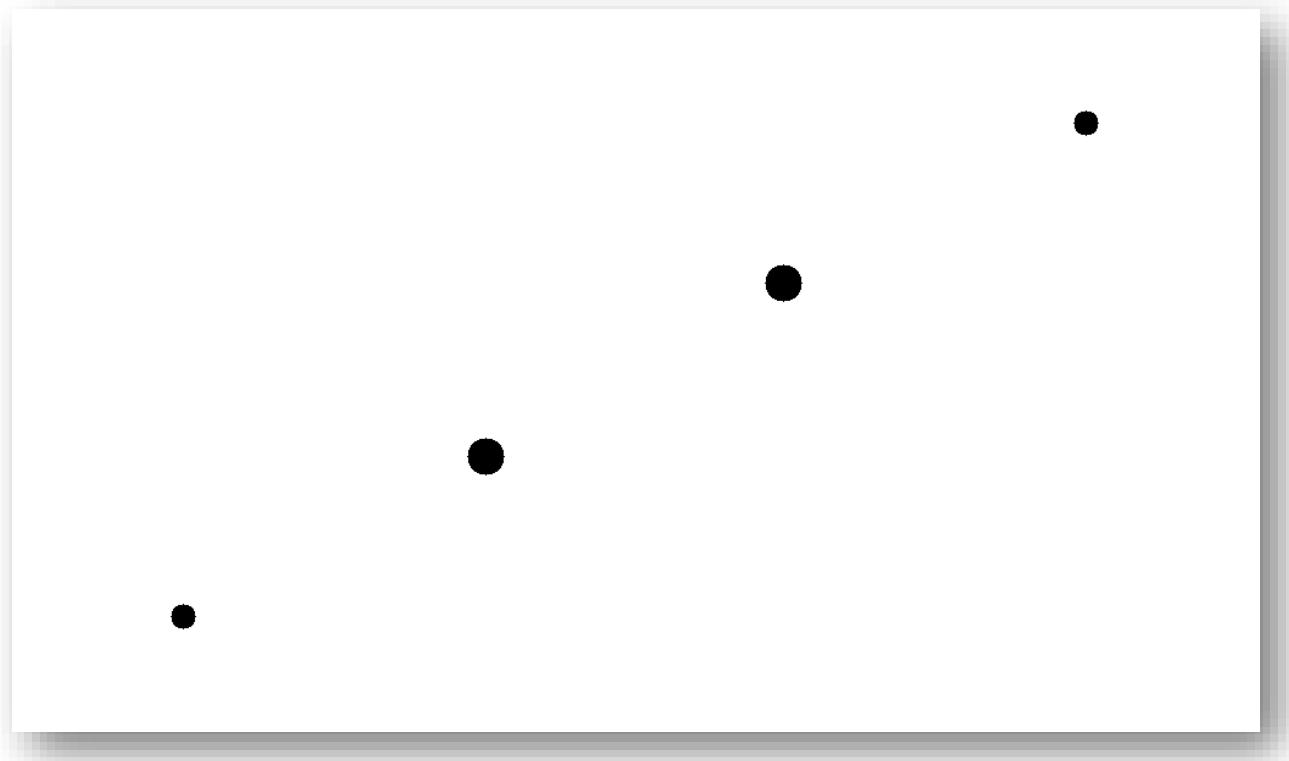
تصویر ورودی:



:p6b\_spectrum.png تصویر ورودی Spectrum



فیلتر ایجاد شده (p6b\_filter.png):



تصویر حاصل از اعمال فیلتر (p6b\_filtered\_image.png):





همین طور که مشاهده می‌شود، نویزهای پریودی حذف شده‌اند.

### قسمت ۵

کد این قسمت در p6c.m قرار دارد. در این قسمت ابتدا تصویر ورودی را می‌خوانیم. سپس آن را به دیتاپیپ double تبدیل می‌کنیم. سپس با استفاده از تابع fftshift و fft2 ضمن انتقال مرکز مختصات به مرکز صفحه تبدیل فوریه‌ی تصویر را می‌گیریم. سپس spectrum و phase angle را ترسیم می‌کنیم.

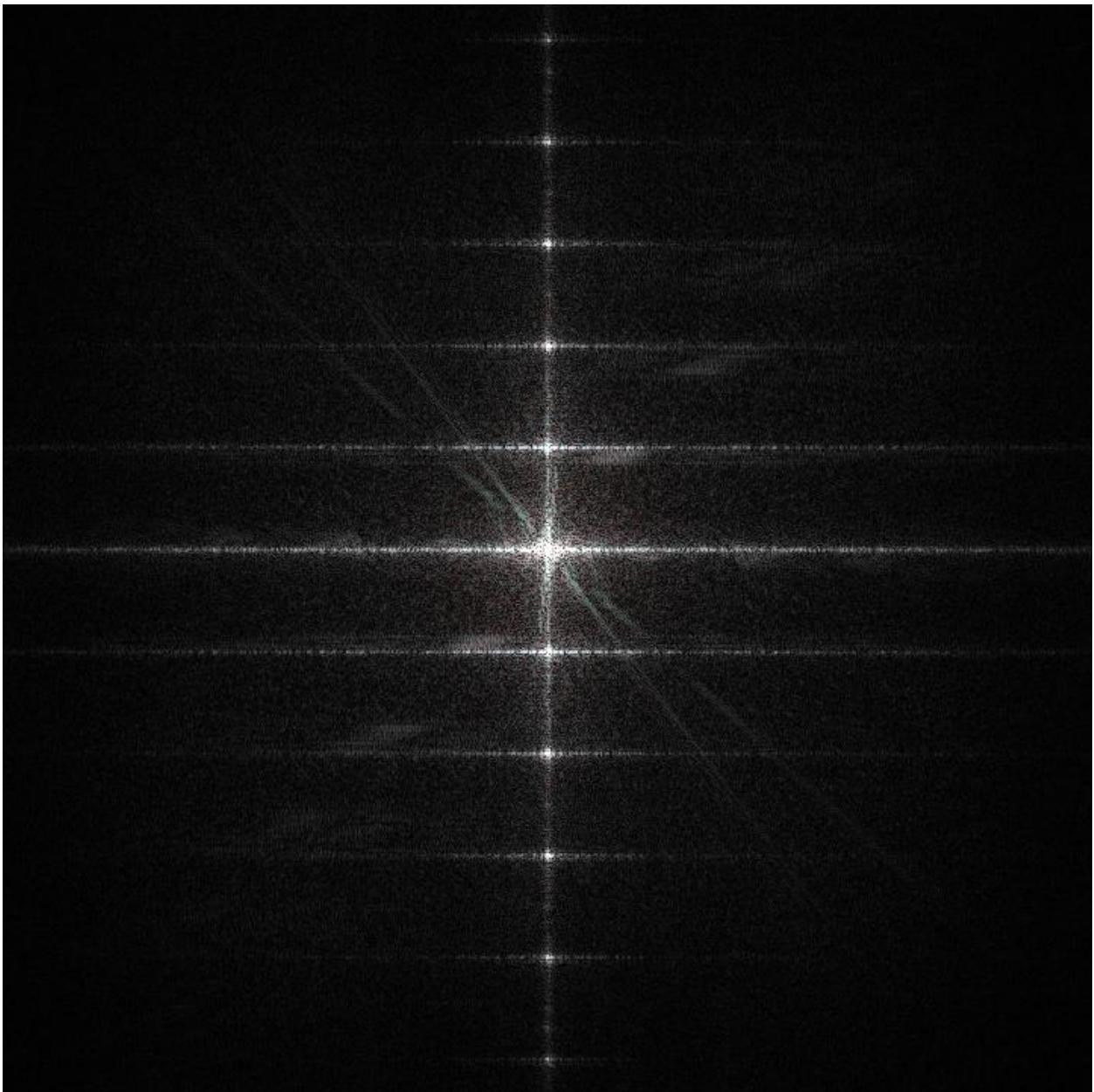
در این قسمت peak یک سری spectrum بر روی خط عمود که از مرکز عبور می‌کند دیده می‌شود که با استفاده از Notch فیلترهای مستطیلی فیلتری برای برطرف کردن آن‌ها ایجاد می‌کنیم. ناج فیلتر می‌توان هر شکلی باشد، مربع، مستطیل، دایره، بر اساس گفته‌ی کتاب). وجود این پیک‌ها به دلیل خطاهای افقی درون تصویر قابل پیش‌بینی بود.

در ادامه با ضرب فیلتر در تبدیل فوریه و تبدیل فوریه‌ی معکوس تصویر فیلتر شده را در حوزه‌ی spatial ایجاد می‌کنیم. در زیر خروجی‌های کد را مشاهده می‌کنید:

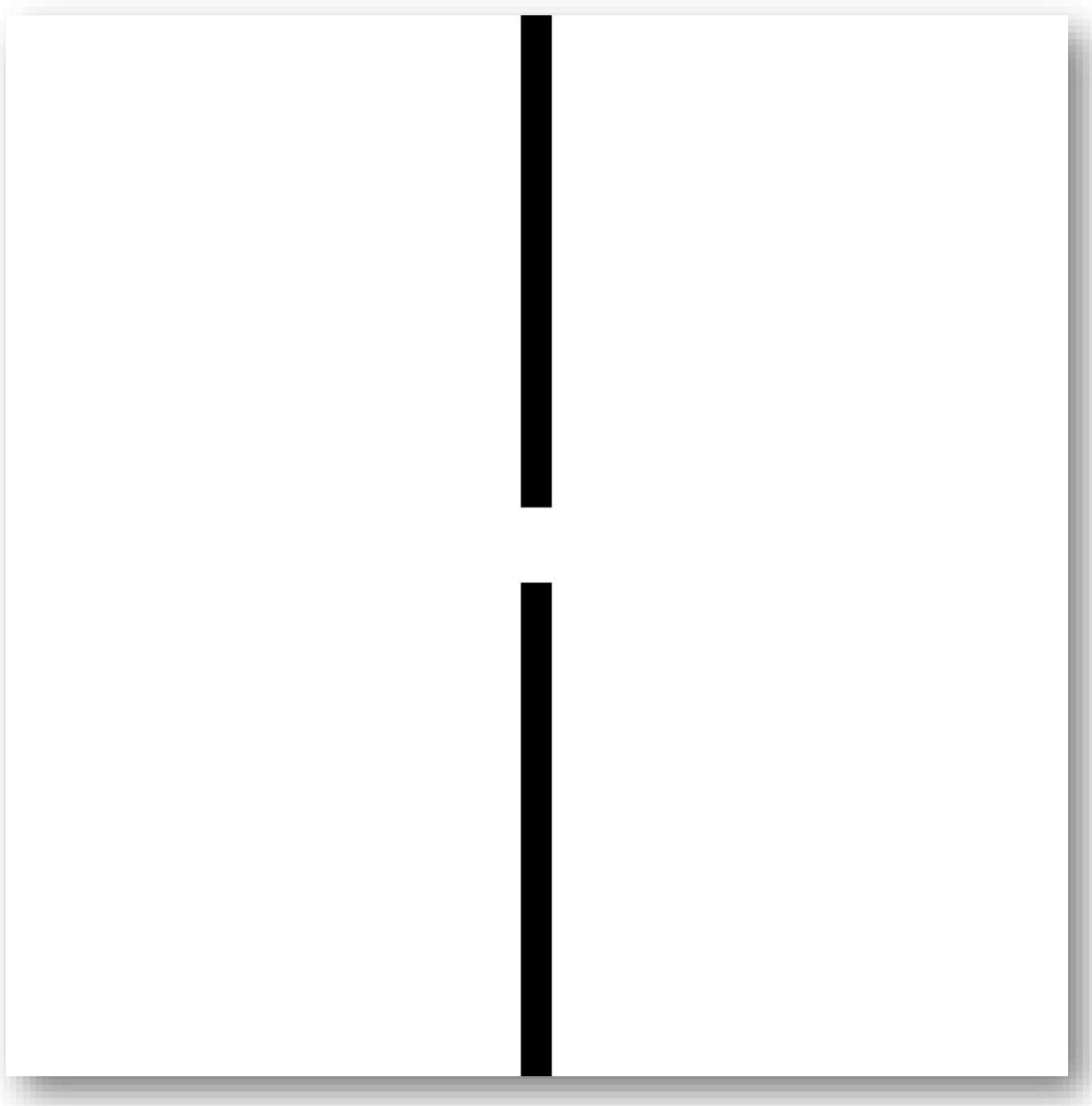
تصویر ورودی:



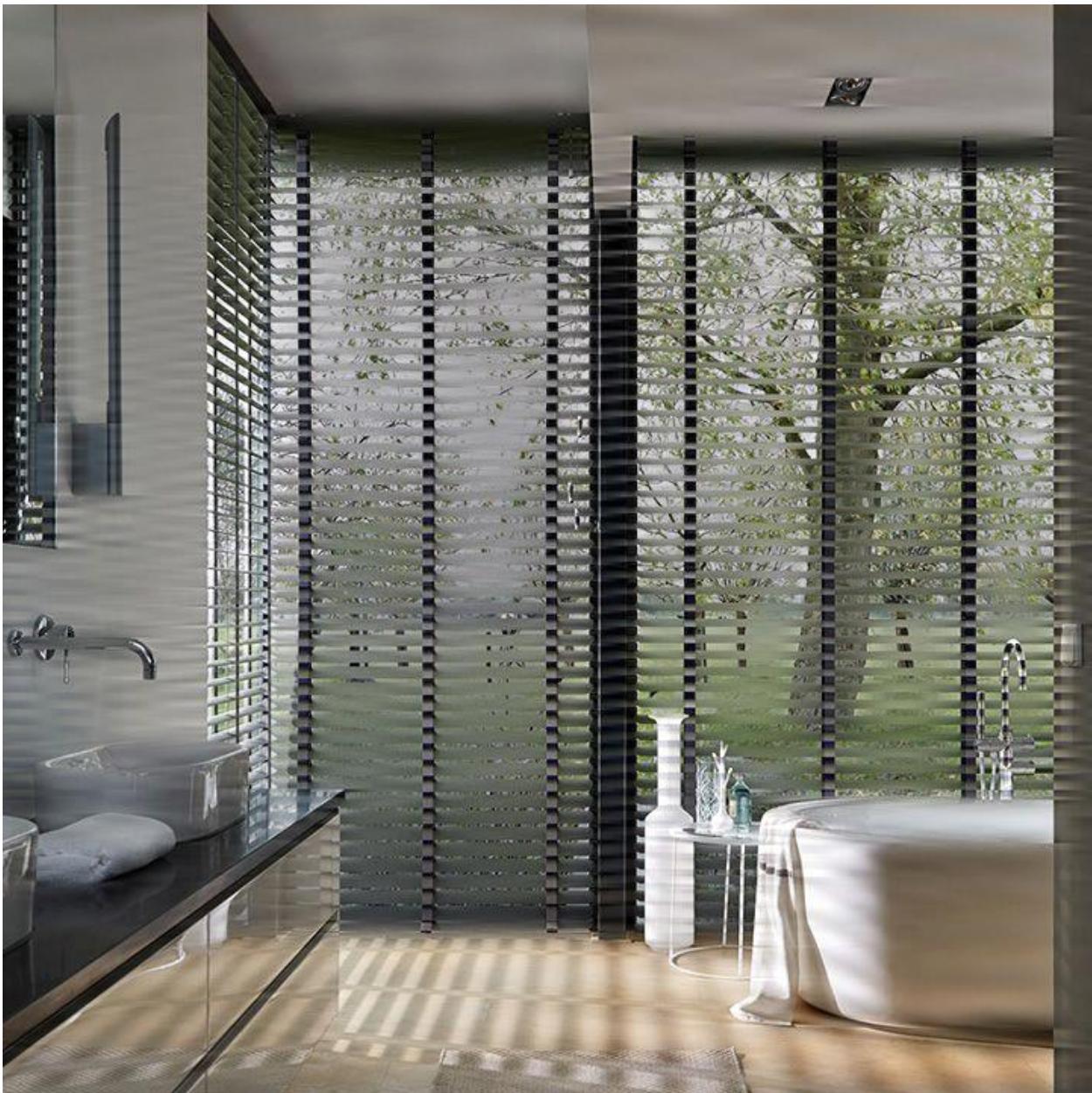
:p6c\_spectrum.png تصویر ورودی Spectrum



فیلتر ایجاد شده (p6c\_filter.png)



تصویر حاصل از اعمال فیلتر : (p6c\_filtered\_image.png)



\* همین طور که مشاهده می شود، در تصویر نویز پریودی وجود نداشت و بخشی از صحنه در spectrum نقاط peak ایجاد می کرد که با استفاده از فیلتر آن را بر طرف کردیم و همین باعث شد میله های افقی درون تصویر(پترن ها) را از بین ببریم.

## قسمت d

کد این قسمت در p6d.m قرار دارد. در این قسمت ابتدا تصویر ورودی را می‌خوانیم. سپس آن را به دیتابیپ تبدیل می‌کنیم. سپس با استفاده از تابع fftshift و fft2 ضمن انتقال مرکز مختصات به مرکز صفحه تبدیل فوریه‌ی تصویر را می‌گیریم. سپس spectrum و phase angle را ترسیم می‌کنیم.

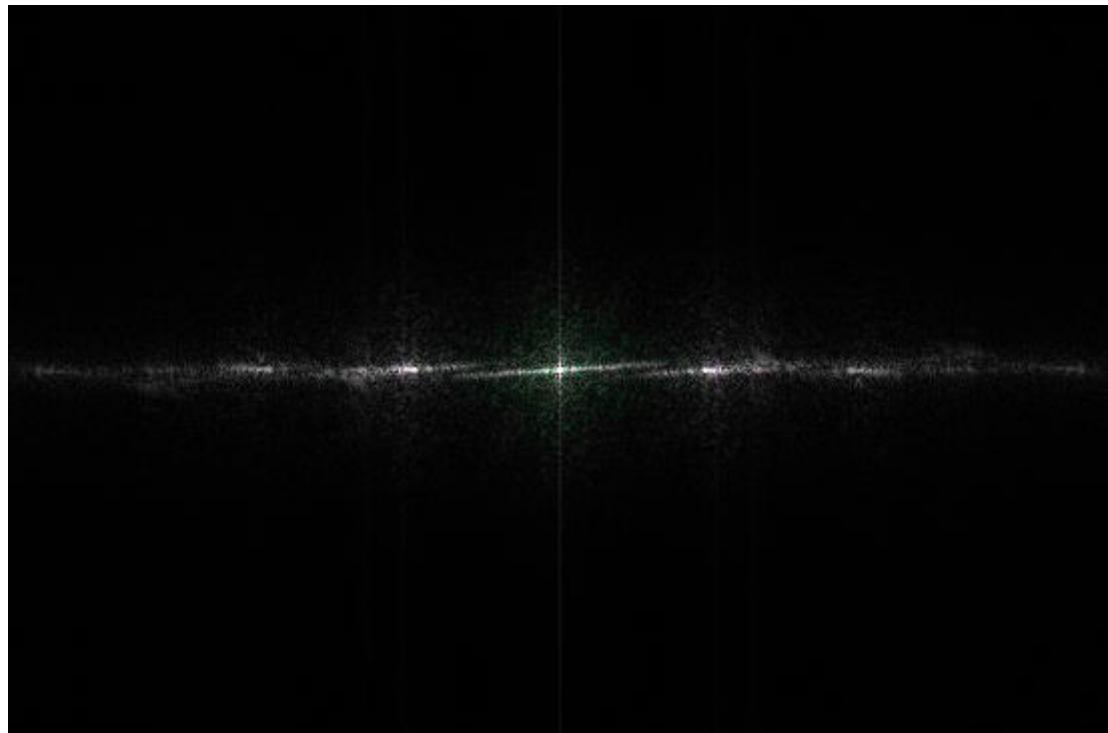
در spectrum یک سری peak بر روی خط افقی که از مرکز عبور می‌کند دیده می‌شود که با استفاده از Notch فیلترهای مستطیلی فیلتری برای برطرف کردن آن‌ها ایجاد می‌کنیم. ناج فیلتر می‌توان هر شکلی باشد، مربع، مستطیل، دایره، بر اساس گفته‌ی کتاب). وجود این پیک‌ها به دلیل خطاهای افقی درون تصویر قابل پیش‌بینی بود.

در ادامه با ضرب فیلتر در تبدیل فوریه و تبدیل فوریه‌ی معکوس تصویر فیلتر شده را در حوزه‌ی spatial ایجاد می‌کنیم. در زیر خروجی‌های کد را مشاهده می‌کنید:

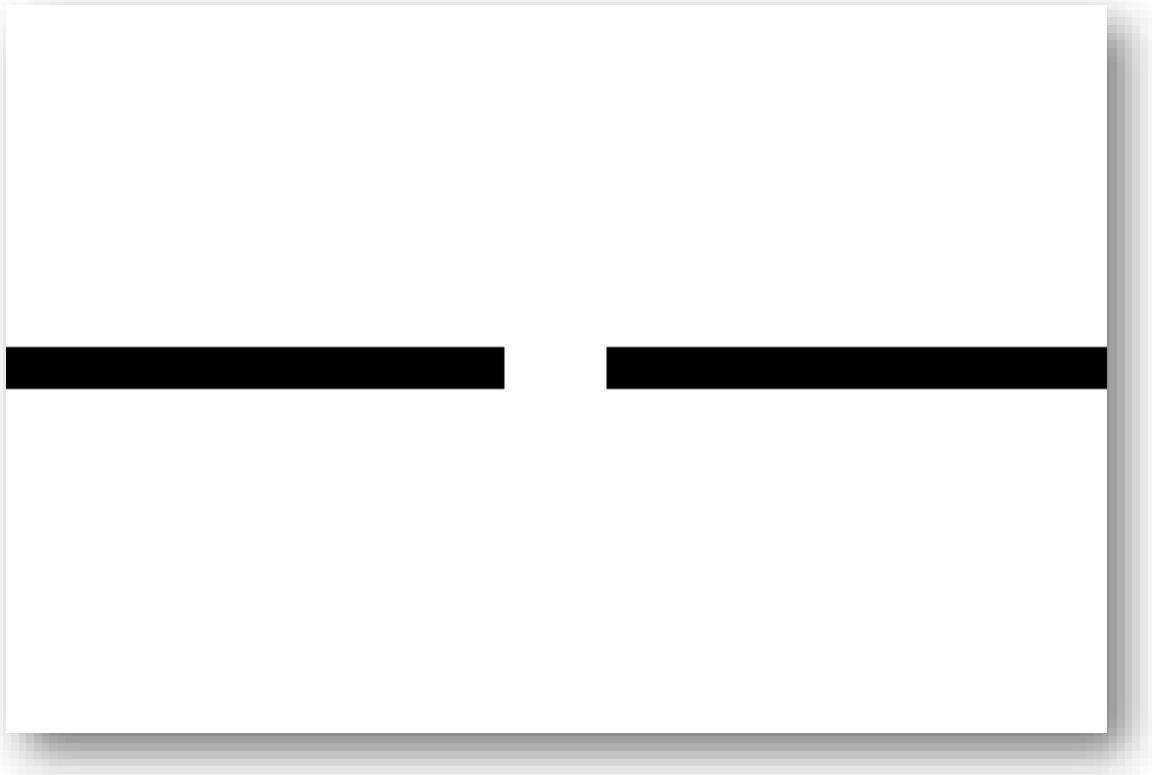
تصویر ورودی:



Spectrum تصویر ورودی : (p6d\_spectrum.png)



فیلتر ایجاد شده (p6d\_filter.png)



تصویر حاصل از اعمال فیلتر (p6d\_filtered\_image.png)



## قسمت ۶

کد این قسمت در p6e.m قرار دارد. در این قسمت ابتدا تصویر ورودی را می‌خوانیم. سپس آن را به دیتاباپ double تبدیل می‌کنیم. سپس با استفاده از تابع fft2 و fftshift ضمن انتقال مرکز مختصات به مرکز صفحه تبدیل فوریه‌ی تصویر را می‌گیریم. سپس spectrum و phase angle را ترسیم می‌کنیم.

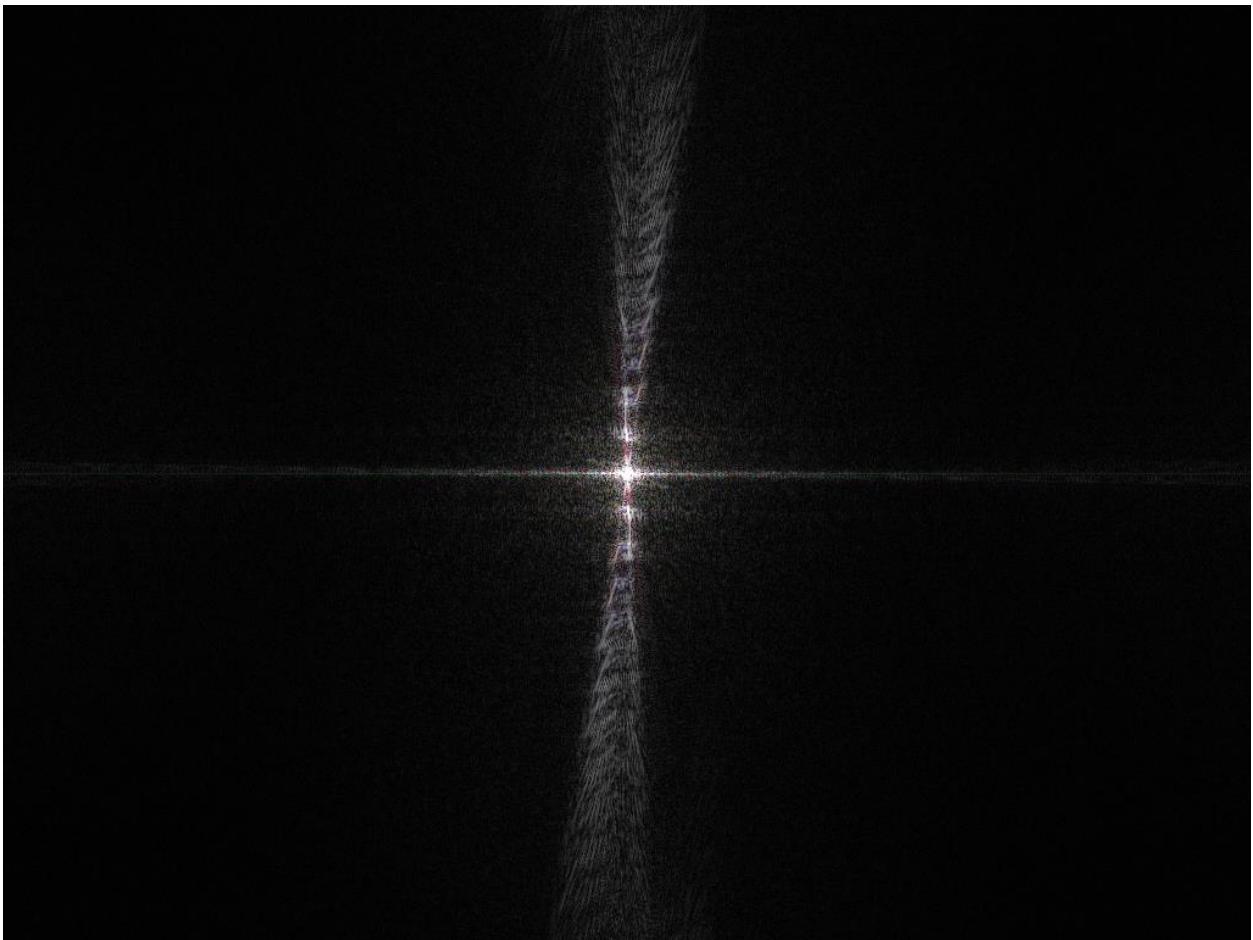
در spectrum یک سری peak بر روی خط عمودی و نزدیک به عمودی که از مرکز عبور می‌کند دیده می‌شود که با استفاده از Notch فیلترهای مستطیلی فیلتری برای برطرف کردن آن‌ها ایجاد می‌کنیم. (ناج فیلتر می‌توان هر شکلی باشد، مربع، مستطیل، دایره، بر اساس گفته‌ی کتاب). وجود این پیک‌ها به دلیل خط‌های تقریباً افقی درون تصویر قابل پیش‌بینی بود.

در ادامه با ضرب فیلتر در تبدیل فوریه و تبدیل فوریه‌ی معکوس تصویر فیلتر شده را در حوزه‌ی spatial ایجاد می‌کنیم. در زیر خروجی‌های کد را مشاهده می‌کنید:

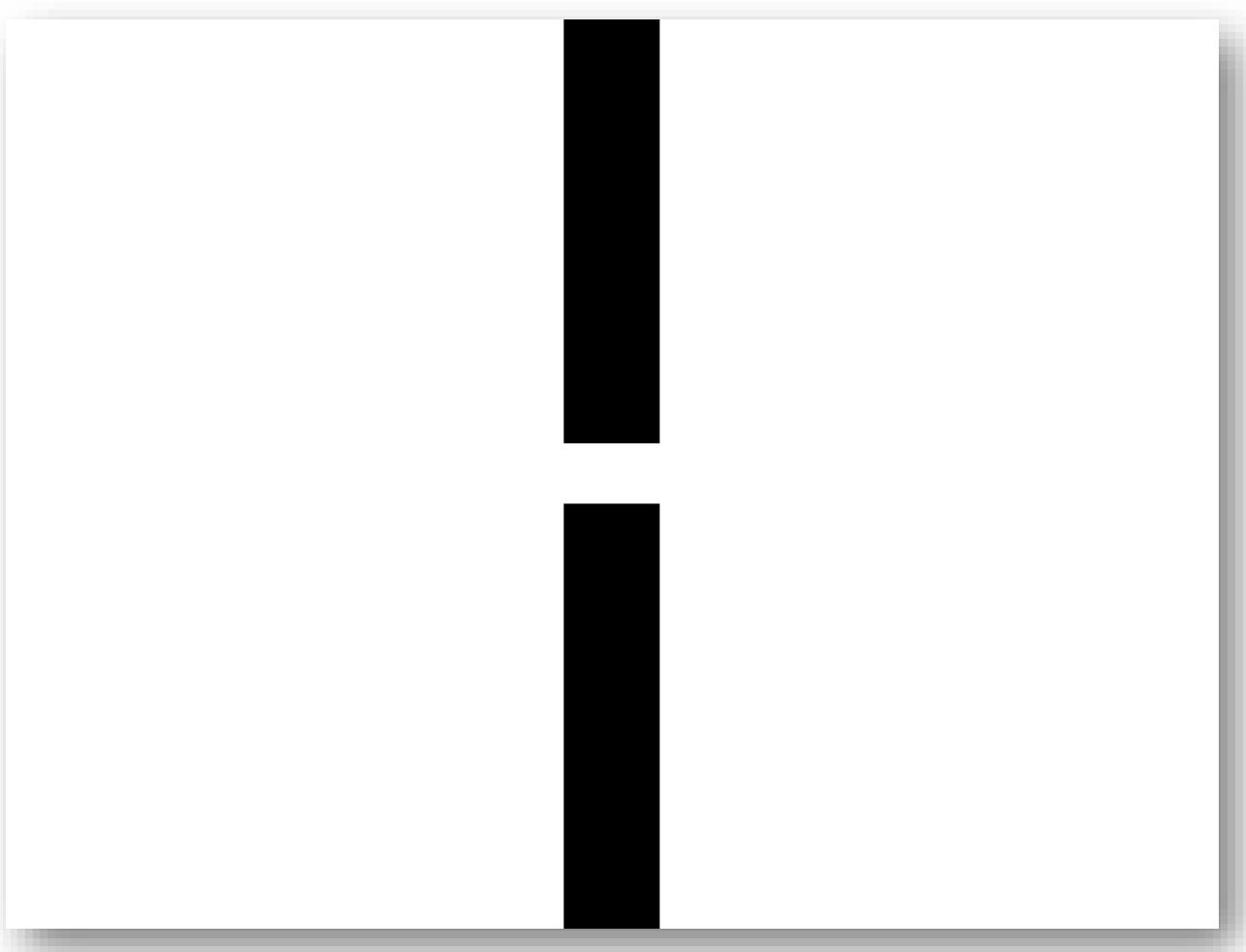
تصویر ورودی:



:p6e\_spectrum.png تصویر ورودی Spectrum



فیلتر ایجاد شده (p6e\_filter.png)



تصویر حاصل از اعمال فیلتر (p6e\_filtered\_image.png)



\* همین طور که مشاهده می‌شود، در تصویر نویز پریودی وجود نداشت و بخشی از صحنه در spectrum نقاط peak ایجاد می‌کرد که با استفاده از فیلتر آن را بر طرف کردیم و همین باعث شد میله‌های تقریباً افقی درون تصویر(پترن‌ها) را از بین ببریم.

#### قسمت f

کد این قسمت در p6f.m قرار دارد. در این قسمت ابتدا تصویر ورودی را می‌خوانیم. سپس آن را به دیتابیپ double تبدیل می‌کنیم. سپس با استفاده از تابع fftshift و fft2 ضمن انتقال مرکز مختصات به مرکز صفحه تبدیل فوریه‌ی تصویر را می‌گیریم. سپس spectrum و phase angle را ترسیم می‌کنیم.

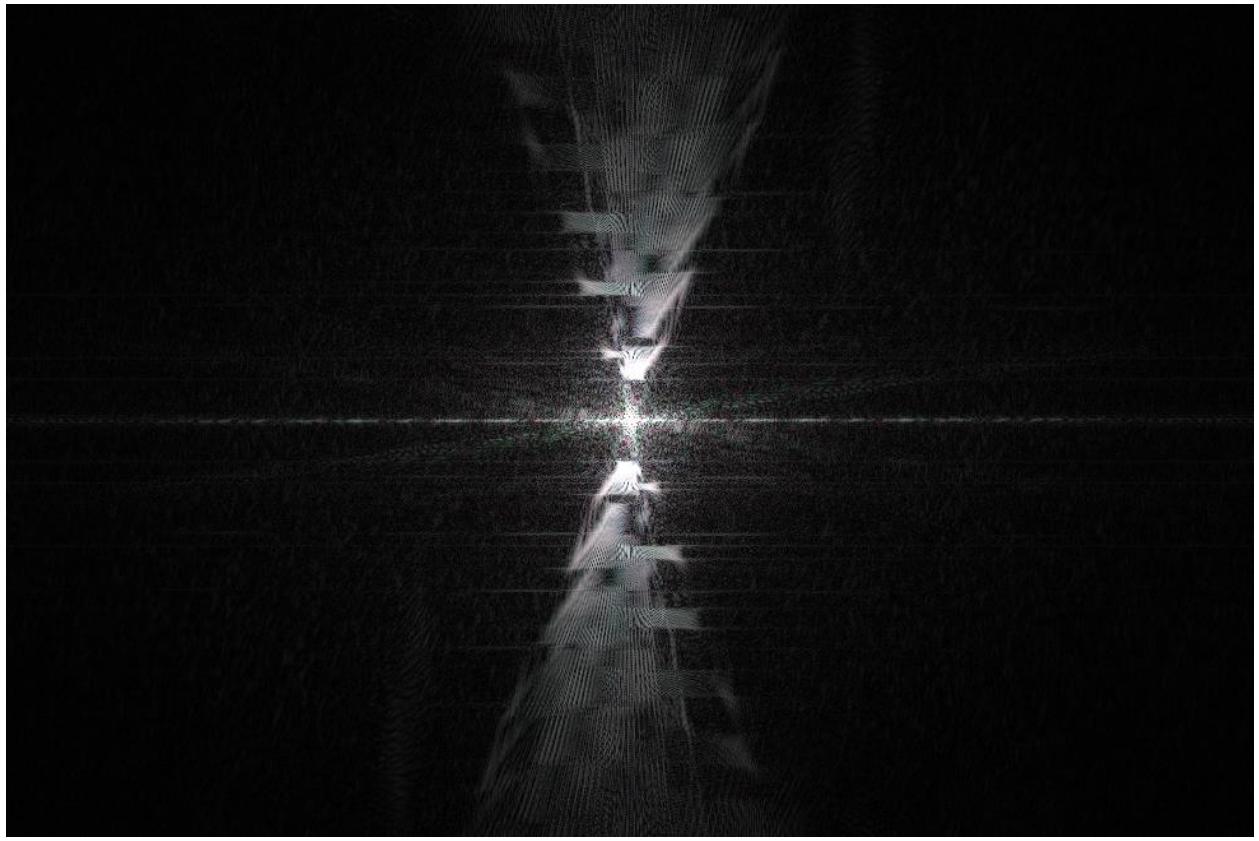
در spectrum یک سری peak بر روی خط عمودی و نزدیک به عمودی که از مرکز عبور می‌کند دیده می‌شود که با استفاده از Notch فیلترهای مستطیلی فیلتری برای برطرف کردن آن‌ها ایجاد می‌کنیم. (ناج فیلتر می‌توان هر شکلی باشد، مربع، مستطیل، دایره، بر اساس گفته‌ی کتاب). وجود این پیک‌ها به دلیل خط‌های تقریباً افقی درون تصویر قابل پیش‌بینی بود.

در ادامه با ضرب فیلتر در تبدیل فوریه و تبدیل فوریه‌ی معکوس تصویر فیلتر شده را در حوزه‌ی spatial ایجاد می‌کنیم. در زیر خروجی‌های کد را مشاهده می‌کنید:

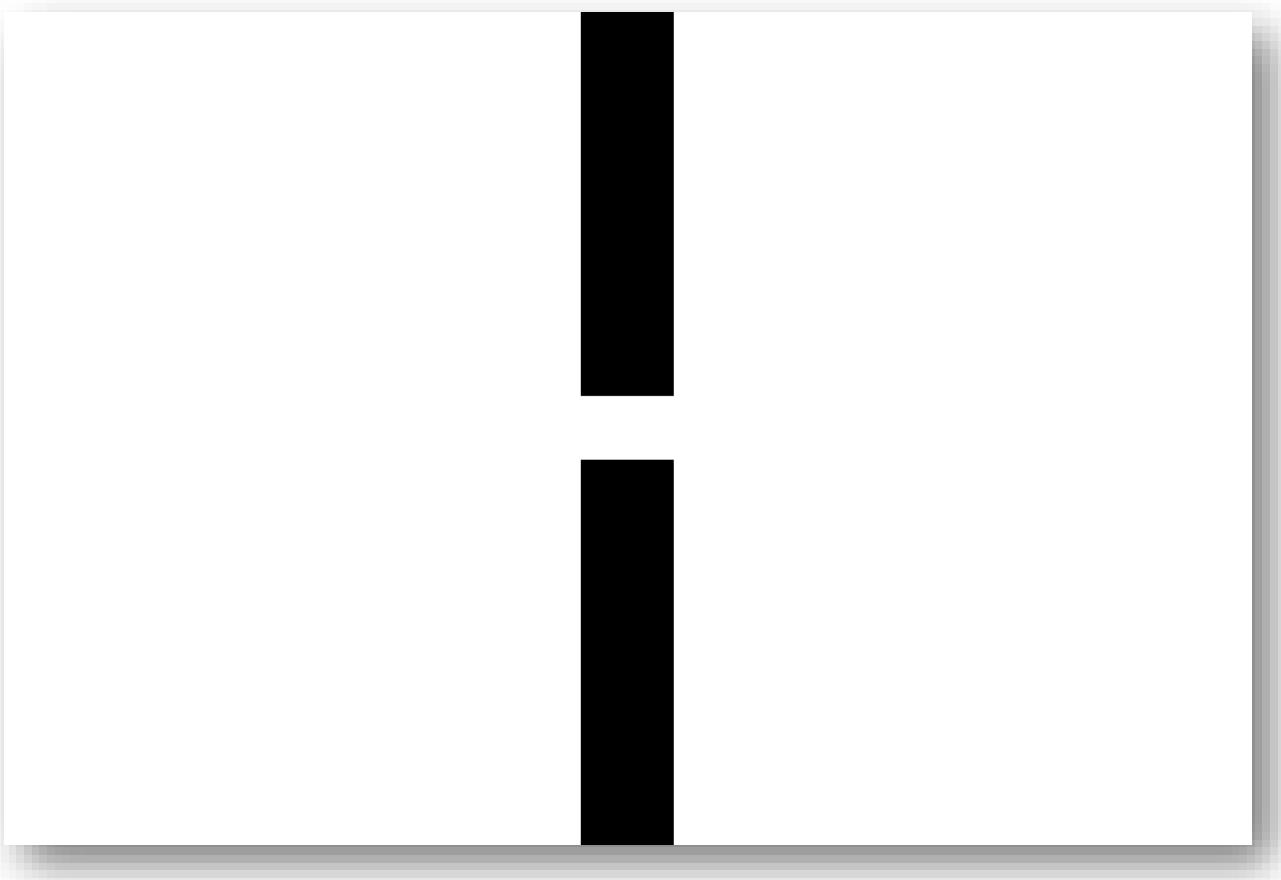
تصویر ورودی:



:p6f\_spectrum.png تصویر ورودی Spectrum



فیلتر ایجاد شده (p6f\_filter.png)



تصویر حاصل از اعمال فیلتر (p6f\_filtered\_image.png)



\* همین طور که مشاهده می شود، در تصویر نویز پریودی وجود نداشت و بخشی از صحنه در spectrum نقاط ایجاد می کرد که با استفاده از فیلتر آن را بر طرف کردیم و همین باعث شد قسمت های تقریباً افقی درون تصویر (پترن ها) را از بین ببریم.

## تمرین ۷

کدهای این قسمت در P7 قرار دارد.

### قسمت a)

کدهای این قسمت در p7a.m قرار دارد. ابتدا تصویر را می‌خوانیم سپس بر اساس مشخصات ارائه شده در صورت سوال فیلتر را در حوزه‌ی spatial ایجاد می‌کنیم. در ادامه تصویر و فیلتر را Pad می‌کنیم. سپس فیلتر و تصویر را به حوزه‌ی فرکانس می‌بریم و بر اساس تابع زیر، فیلتر wiener را اعمال می‌کنیم:

$$\hat{F}(u, v) = \left[ \frac{1}{H(u, v)} \frac{|H(u, v)|^2}{|H(u, v)|^2 + K} \right] G(u, v)$$

در انتهای، تصویر فیلتر شده با استفاده از تبدیل معکوس فوریه به حوزه‌ی Spatial برمی‌گردانیم. در زیر خروجی‌های کد را مشاهده می‌کنید:

تصویر ورودی:



تصویر فیلتر شده  $K$  برابر است با  $\cdot, \cdot 2$  : (p7a\_filtered\_image.png)



### قسمت b

کدهای این قسمت در p7b.m قرار دارد. ابتدا تصویر را می‌خوانیم سپس بر اساس مشخصات ارائه شده در صورت سوال فیلتر را در حوزه‌ی spatial ایجاد می‌کنیم. در ادامه تصویر و فیلتر را Pad می‌کنیم. سپس فیلتر و تصویر را به حوزه‌ی فرکانس می‌بریم و بر اساستابع زیر، فیلتر wiener را اعمال می‌کنیم:

$$\hat{F}(u, v) = \left[ \frac{1}{H(u, v)} \frac{|H(u, v)|^2}{|H(u, v)|^2 + K} \right] G(u, v)$$

در انتها، تصویر فیلتر شده با استفاده از تبدیل معکوس فوریه به حوزه‌ی Spatial برمی‌گردانیم. در زیر خروجی‌های کد را مشاهده می‌کنید:

تصویر ورودی:



تصویر فیلتر شده K برابر است با (p7a\_filtered\_image.png) .۰۲



## تمرین ۸

کدهای این قسمت در p8b قرار دارد.

### قسمت a

کد این قسمت از سوال در p8a.m قرار دارد. برای این سوال فیلترها و روش‌های بسیار مختلفی را چک کردم. تصویر بسیار تخریب شده است. در ابتدا تصویر را رسم می‌کنم و بعد یک قسمت یکنواخت از تصویر را انتخاب می‌کنم و هیستوگرام آن را رسم می‌کنم:

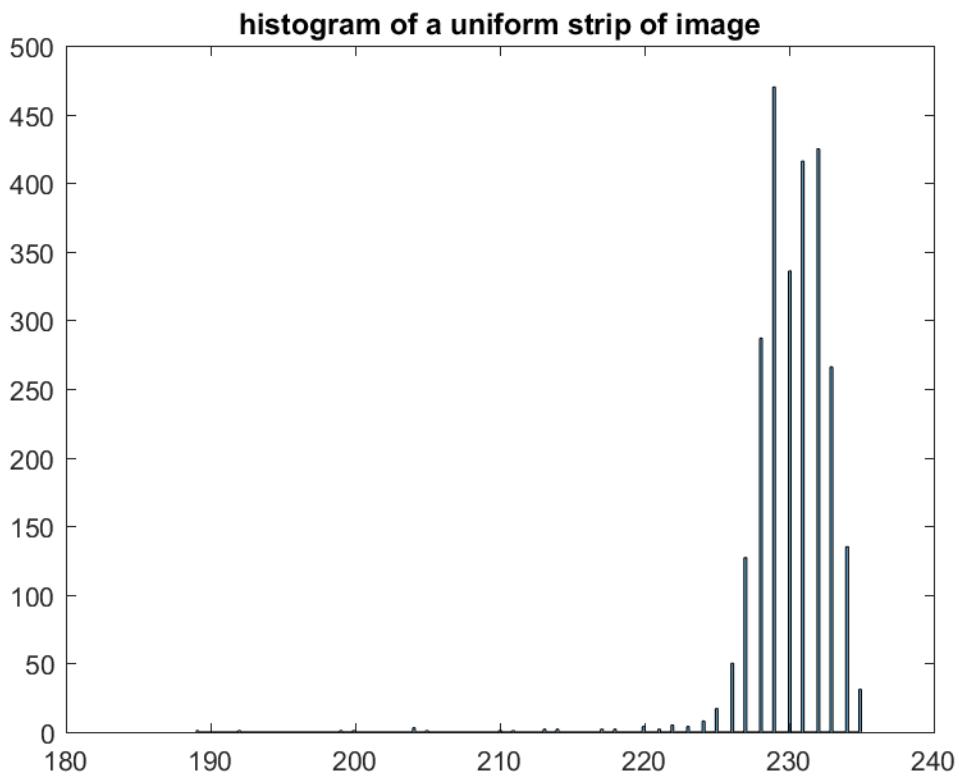
تصویر ورودی:



قسمت یکنواخت:



هیستوگرام قسمت یکنواخت:



همین طور که مشاهده می شود، نقاط با شدت روشنایی هایی بسیار زیاد و یا بسیار کم دیده می شود همین طور نویزی با حالت gaussian موجود است. به همین دلیل ابتدا از فیلتر میانه استفاده می کنم سپس از equalization و در ادامه از wiener استفاده می کنم.

تصویر بعد از اعمال :() median



تصویر بعد از اعمال  $\text{histeq}()$



تصویر بعد از اعمال : ( weiner )



تصویر  
ورودی



آخرین خروجی



## قسمت b

کد این قسمت از سوال در p8b.m قرار دارد. اگر به تصویر توجه کنیم در آن دو مشکل دیده می‌شود.

- (۱) نقطه‌های سفید رنگ به طور نا یکنواخت در تصویر وجود دارد.
- (۲) یک پترن مانند صفحه‌ی شترنج در تصویر دیده می‌شود.

تصویر ورودی:

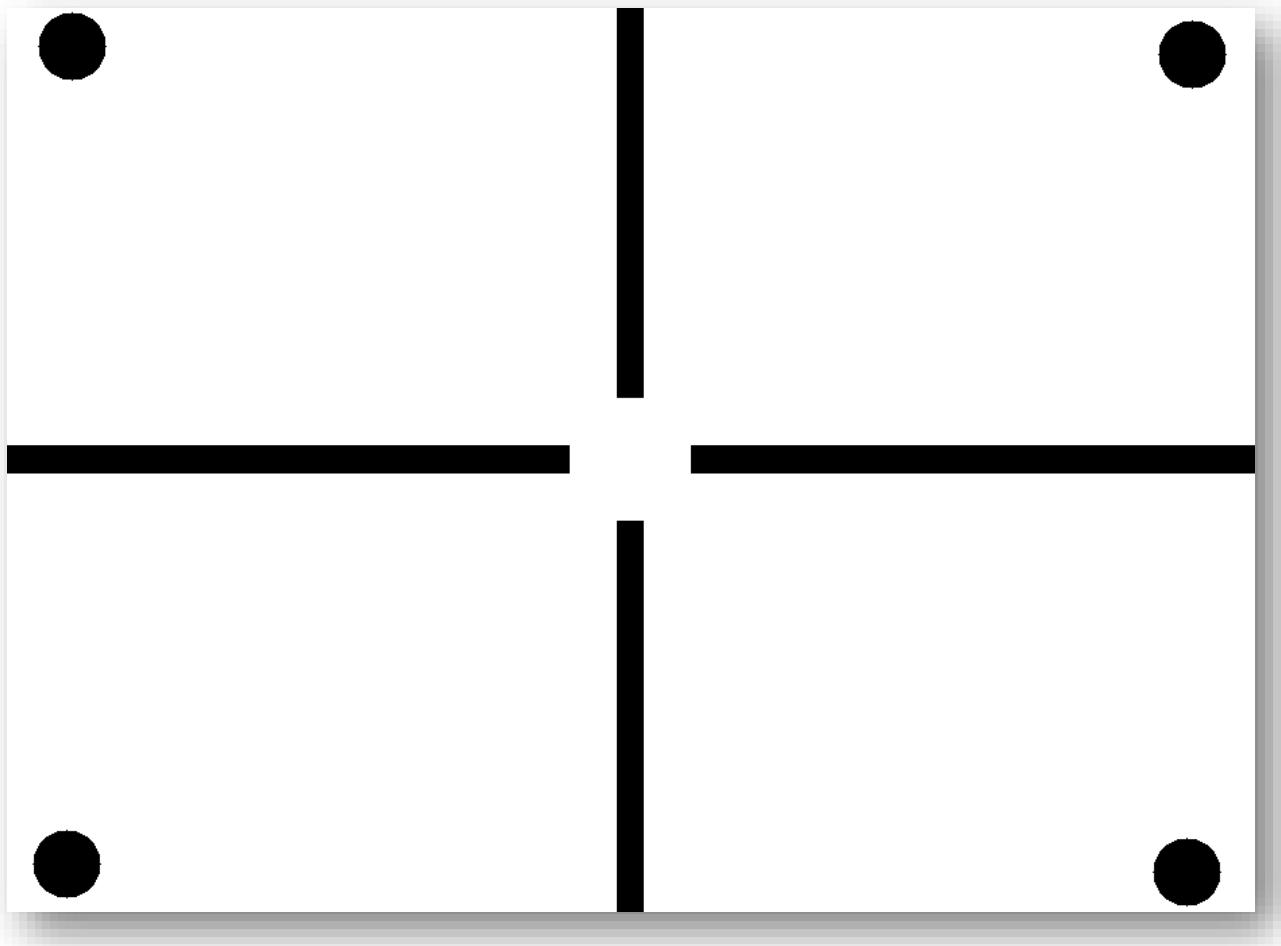


در ابتدا با استفاده از فیلتر minimum با اندازه‌ی  $3 \times 3$  نقطه‌های سفید را حذف می‌کنیم:(p8b\_filtered\_1.png)



در ادامه با استفاده از ۴ فیلتر ناچ (دو عدد دایره شکل، دو عدد مستطیل شکل) پترن موجود در تصویر را حذف می‌کنیم، در زیر خروجی حاصل از این کار را مشاهده می‌کنید:

تصویر فیلتر استفاده شده (p8b\_\_fourier\_filter.png)



تصویر حاصل از فیلتر (p8b\_filtered\_2.png)



تصویر ورودی و آخرین خروجی:

تصویر  
ورودی



آخرین خروجی



### قسمت C

کد این قسمت از سوال در p8c.m قرار دارد. اگر به تصویر توجه کنیم در آن دو مشکل دیده می‌شود.

- (۱) نقطه‌هایی سیاه رنگ به طور نا یکنواخت در تصویر وجود دارد.
- (۲) خطهایی سیاه مانند اثر فرچه بر روی تصویر وجود دارد.

تصویر ورودی:



در ابتدا با استفاده از فیلتر  $\text{max}$  با اندازه‌ی  $3 \times 3$  نقطه‌های سیاه را حذف می‌کنیم و اثر خطهای سیاه فرچه مانند را تا جای ممکن از بین می‌بریم (p8c\_filtered\_1.png):



در ادامه با استفاده از فیلتر گاووسی با انحراف از معیار  $8,0$  تصویر را کمی بلور می‌کنیم تا به حالت عادی بعد از ماکزیمم گرفتن برگردد و حالت سوپر پیکسلی از بین برود، در زیر خروجی حاصل از این کار را مشاهده می‌کنید:

تصویر حاصل از فیلتر (p8c\_filtered\_2.png)



تصویر ورودی و آخرین خروجی:

تصویر  
ورودی



آخرین خروجی



#### قسمت d

کد این قسمت از سوال در p8d.m قرار دارد. اگر به تصویر توجه کنیم در آن دو مشکل دیده می‌شود.

- (۱) نقطه‌هایی سفید رنگ به طور نا یکنواخت در تصویر وجود دارد.
- (۲) در تصویر خراش‌هایی وجود دارد.
- (۳) یک نوع حالت پارچه مانند از سمت چپ به راست تصویر مشاهده می‌شود.

تصویر ورودی:



در ابتدا با استفاده از فیلتر wiener با اندازه‌ی  $7*7$  تاثیر خراش و حالت بافت پارچه مانند را تا جای ممکن از بین می‌بریم (p8d\_filtered\_0.png)



در ادامه با استفاده از فیلتر `ordfilt2` با اندازه‌ی پنجره‌ی  $4^*4$  و انتخاب درایه سوم، نقاط سفید رنگ را از بین می‌بریم و تاثیر خراش‌ها را کم‌تر می‌کنیم، در زیر خروجی حاصل از این کار را مشاهده می‌کنید:  
`(p8d_filtered_1.png)`



تصویر ورودی و آخرین خروجی:

آخرین خروجی

تصویر

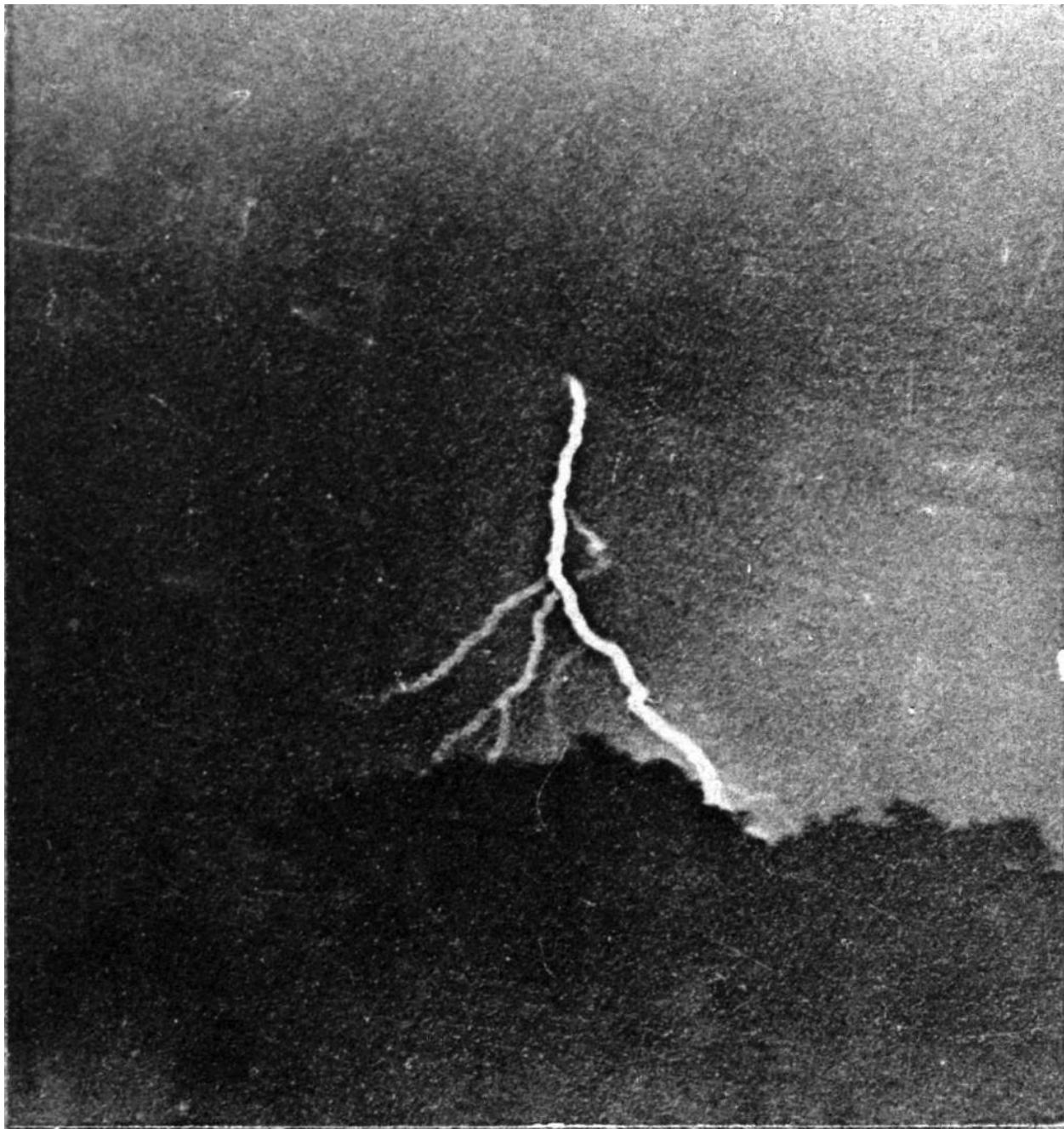
ورودی



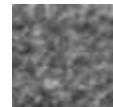
قسمت ۶

کد این قسمت در p8e.m قرار دارد. در ابتدا تصویر را می‌خوانیم سپس یک قسمت یک نواخت از تصویر را انتخاب می‌کنیم و هیستوگرام آن را رسم می‌کنیم. در زیر تصویر اصلی، قسمت انتخاب شده و هیستوگرامش را مشاهده می‌کنید:

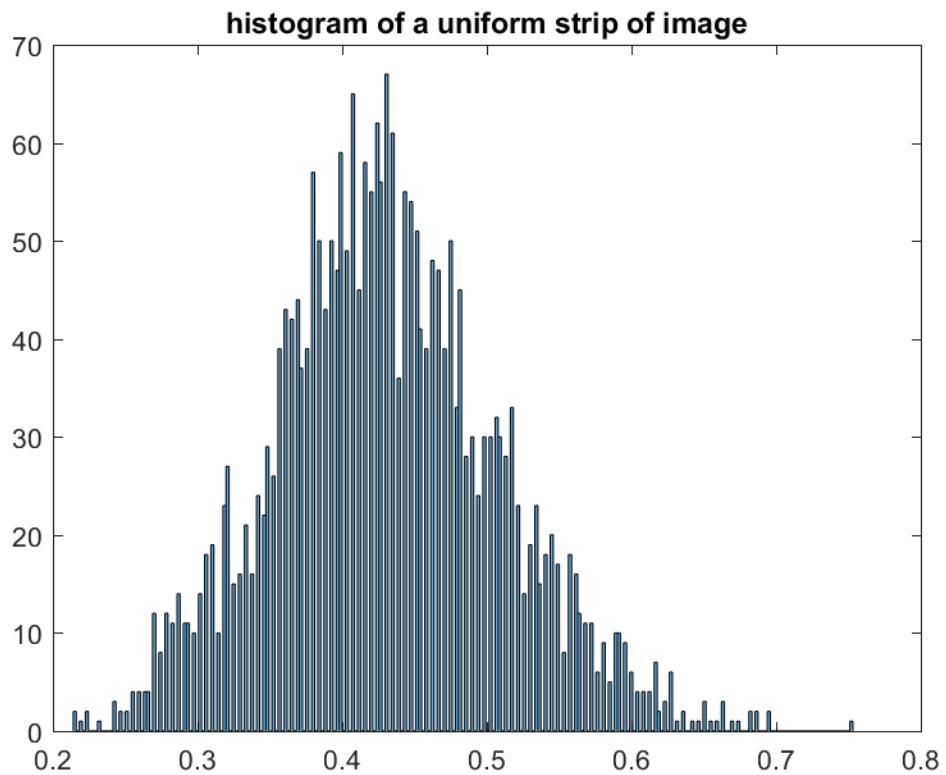
تصویر ورودی:



قسمت یکنواخت انتخاب شده ():



: (p8e\_hist\_strip\_image.png) هیستوگرام قسمت یکنواخت انتخاب شده



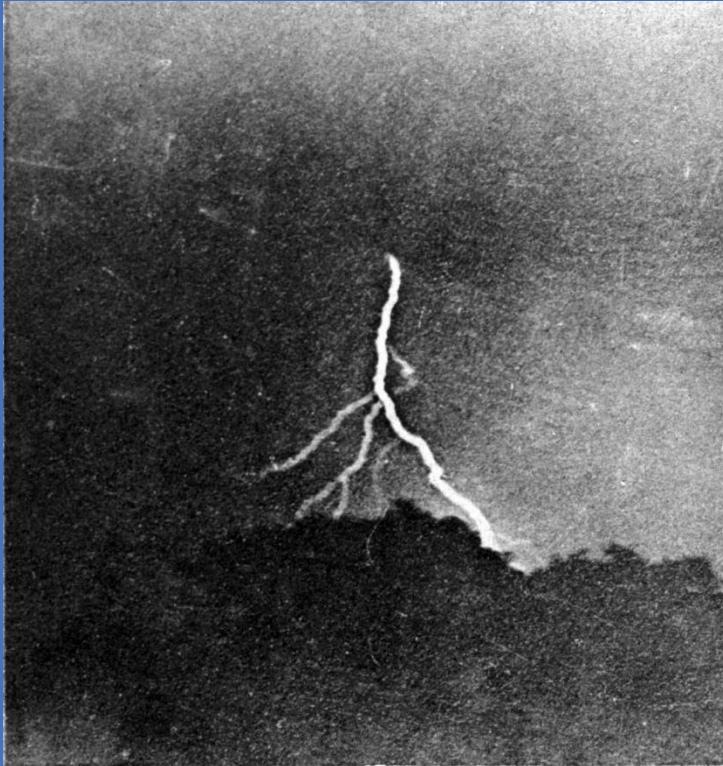
همین که در بالا مشاهده می شود تصویر نویز از نوع Gaussian دارد، به همین دلیل از فیلتر weiner برای رفع آن استفاده می کنیم. تصویر بعد از اعمال فیلتر را مشاهده می کنید (p8e\_filtered\_1.png):



یکسری نقاط سفید هم دیده می شود که برای کمتر کردن شان از فیلتر minimum استفاده می کنم. در زیر خروجی این عمل را مشاهده می کنید : (p8e\_filtered\_2.png)



تصویر  
ورودی



آخرین خروجی



کد این قسمت در p8f.m قرار دارد. ابتدا تصویر را می‌خوانم و نمایش می‌دهم. همین طور که در تصویر دیده می‌شود تخریب‌هایی در شکل به صورت نقاط سفید و سیاه و خراش دیده می‌شود. برای از بین بردن آن‌ها از فیلتر میانه با سایز  $4 \times 4$  استفاده می‌کنم در شکل زیر تصویرهای حاصل از اجرای کد را مشاهده می‌کنید:

تصویر ورودی:



تصویر بعد از اعمال فیلتر میانه :p8f\_filtered\_1.png



تصویر  
ورودی



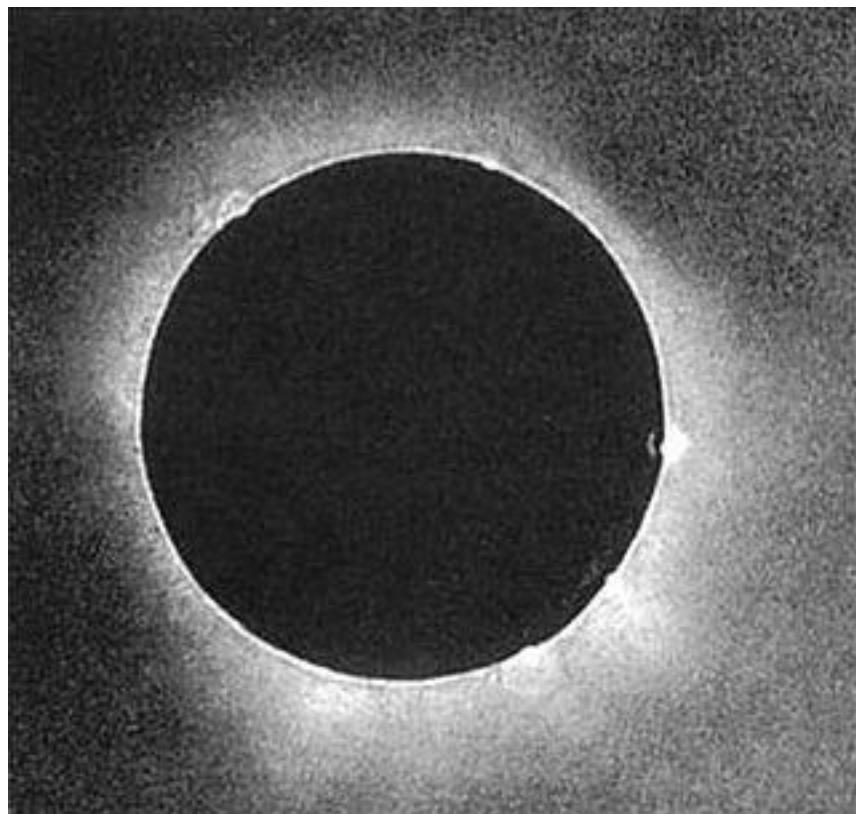
آخرین خروجی



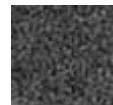
همین طور که مشاهده می شود حجم بسیار زیادی از نقاط سیاه و سفید از بین رفته اند.

کد این قسمت در p8g.m قرار دارد. ابتدا تصویر را می‌خوانم و نمایش می‌دهم. سپس در ادامه یک ناحیه‌ی یکنواخت را انتخاب می‌کنم و هیستوگرام آن را نمایش می‌دهم. در زیر ناحیه‌ی یکنواخت انتخاب شده و هیستوگرامش را مشاهده می‌کنید:

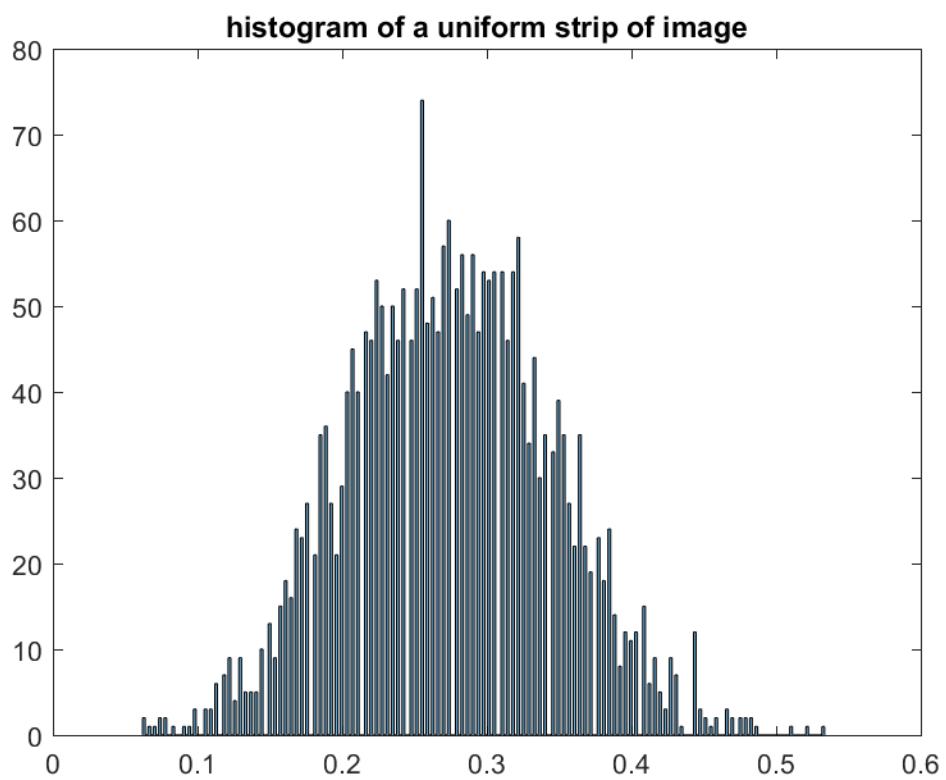
تصویر ورودی:



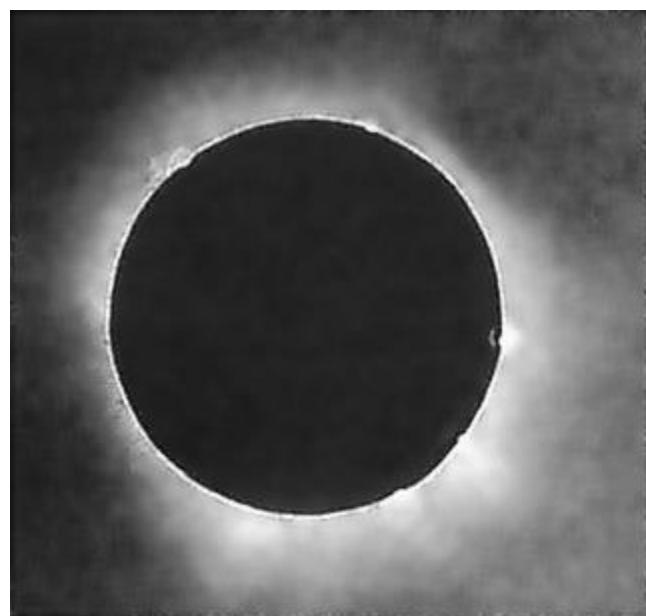
ناحیه‌ی یکنواخت انتخاب شده:



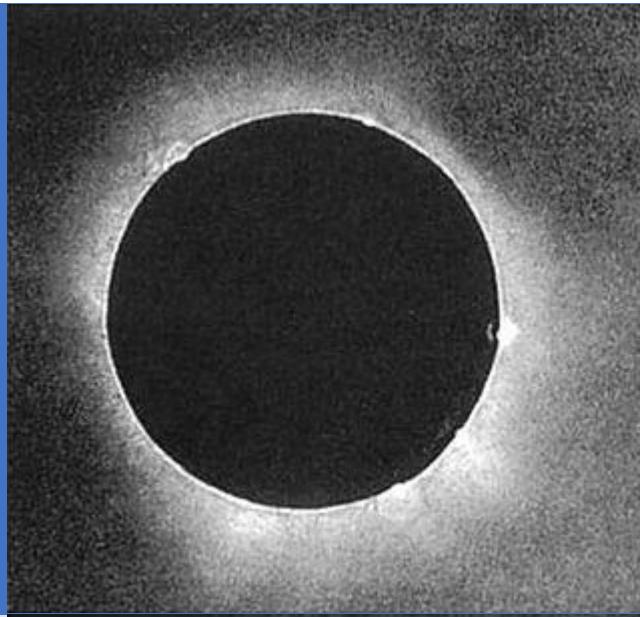
هیستوگرام ناحیه‌ی یکنواخت انتخاب شده:



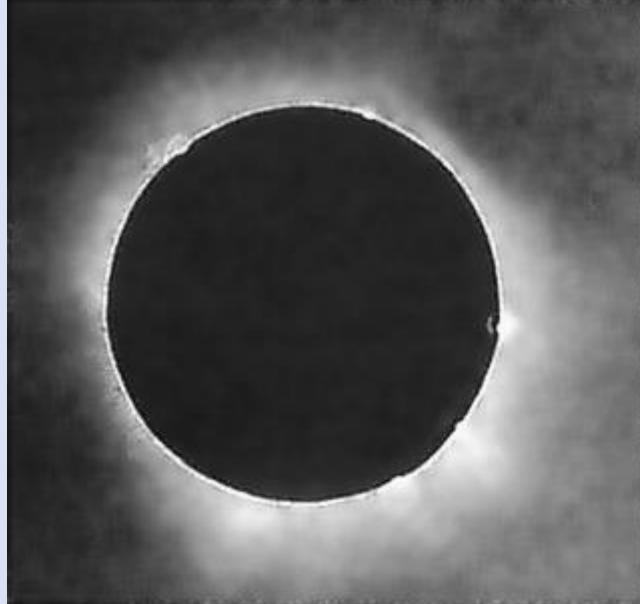
همین طور که مشاهده می شود هیستوگرام این تصویر دارای نویز گاوسی است به همین دلیل از فیلتر wiener برای رفع نویز استفاده می کنم. در زیر خروجی اعمال wiener با سایز  $7 \times 7$  را مشاهده می کنید(p8g\_filtered\_1.png)



تصویر  
ورودی



آخرین خروجی



## قسمت h

کد این قسمت در p8h.m قرار دارد. همان طور که در تصویر دیده می‌شود، نقاط سفید رنگی تصویر را تخریب کرده‌اند همچنین تعداد بسیار کمی نقطه‌ی سیاه دیده می‌شود. به همین دلیل از `ordfilter2` با اندازه‌ی پنجراهی ۳\*۳ استفاده می‌کنم و ۴ امین المنش در ترتیب سورت شده‌ی پیکسل‌های درون همسایگی را انتخاب می‌کنم. همچنین در انتهای تصویر را Sharp می‌کنم.

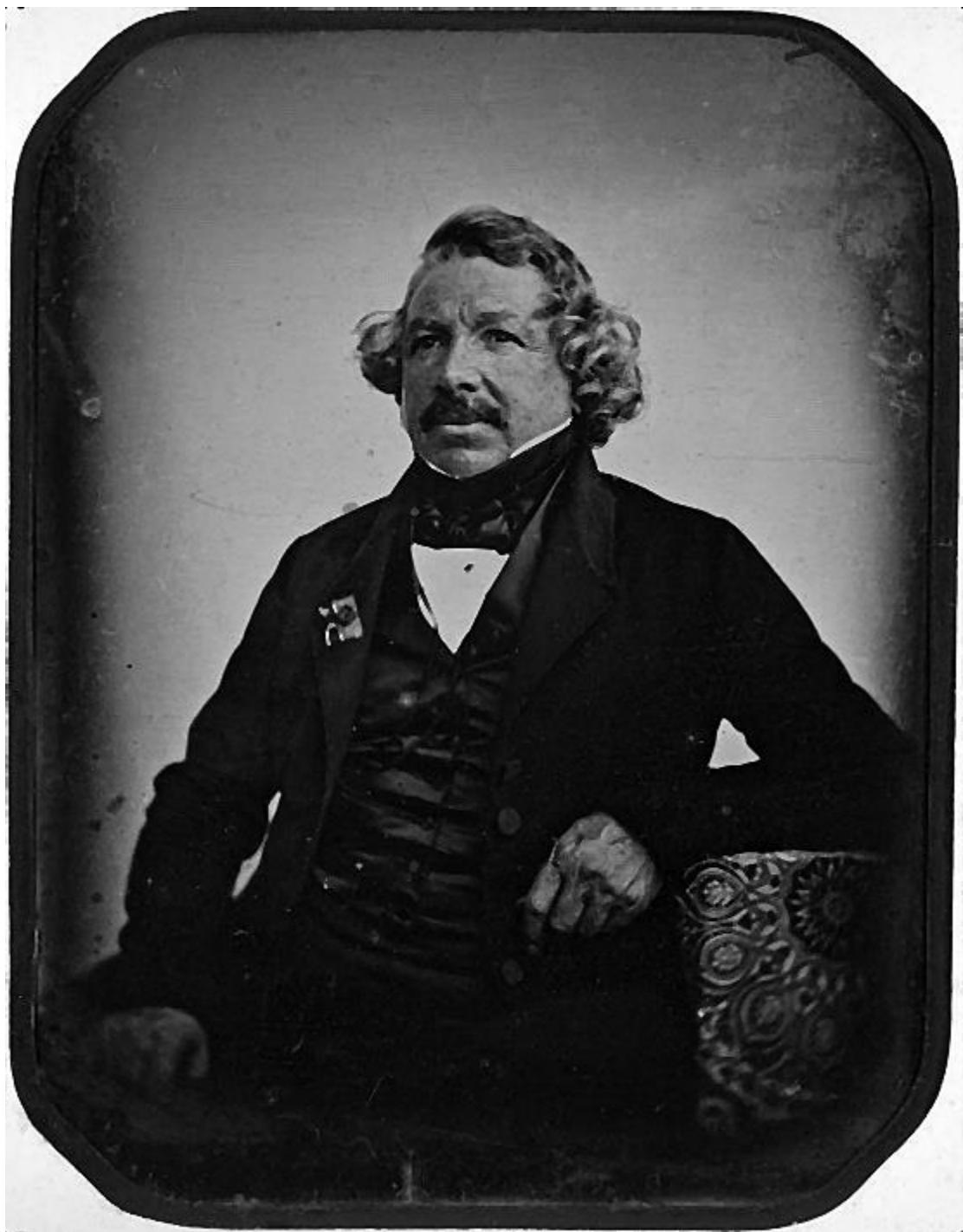
تصویر ورودی:



تصویر بعد از اعمال ordfilter2 : (p8h\_filtered\_1.png)



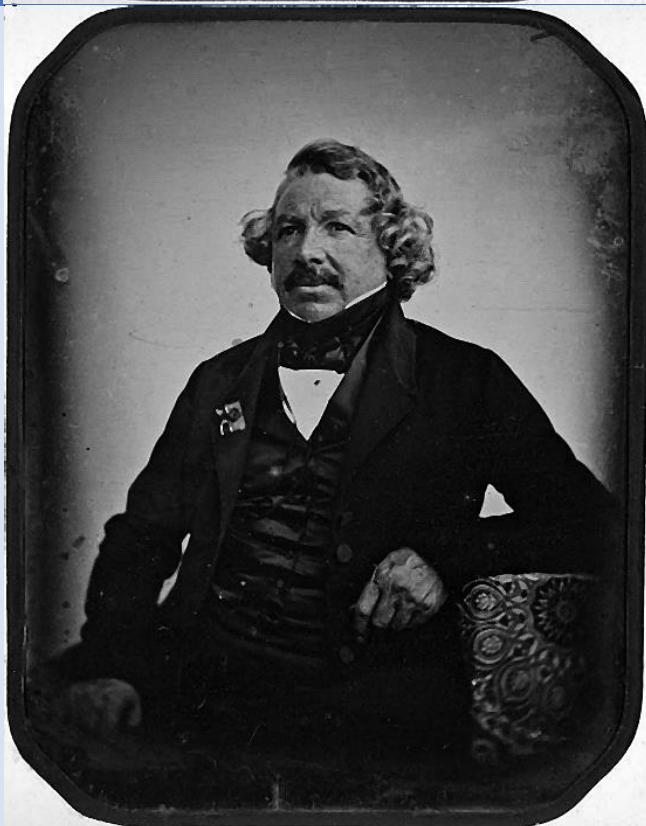
تصویر بعد از شارپ کردن (p8h\_filtered\_2.png)



تصویر  
ورودی



آخرین خروجی



قسمت از

کد این قسمت در p8i.m قرار دارد. در کد ابتدا تصویر را نمایش می‌دهم سپس یک قسمت یکنواخت تصویر را انتخاب می‌کنم و در هیسترام آن را رسم می‌کنم:

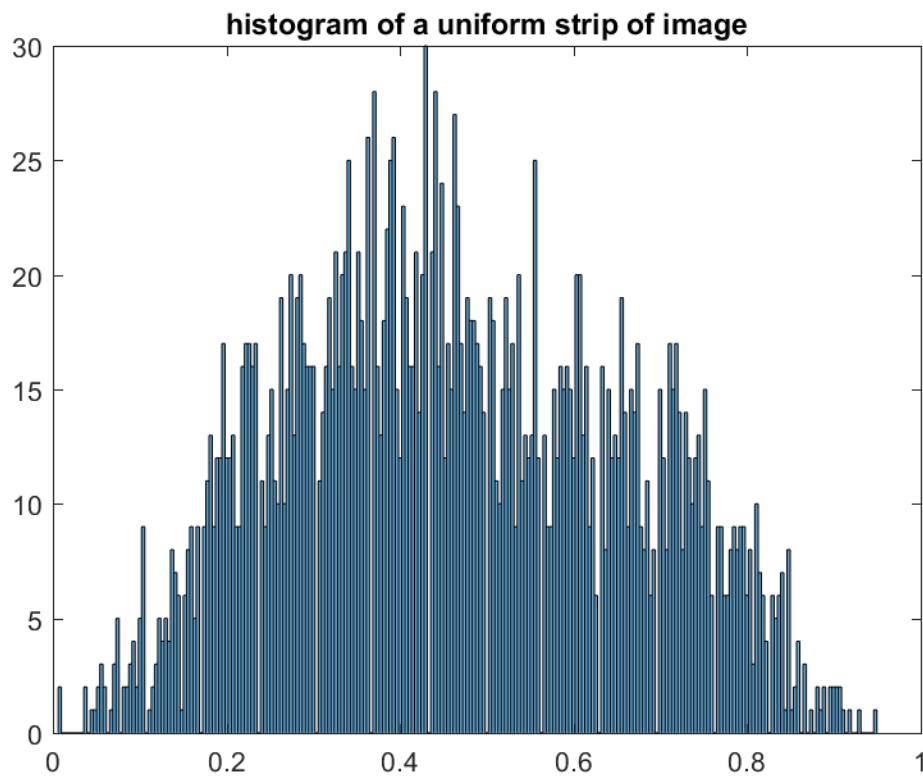
تصویر ورودی:



تصویر قسمت یکنواخت (p8i\_strip.png)



هیستوگرام قسمت یکنواخت (p8i\_hist\_strip\_image.png)



همین طور که دیده می شود رد تصویر نویز گووسی وجود دارد همچنانی نقاط سفید زیادی در تصویر دیده می شود. تعداد نقطه‌ی سیاه هم مشاهده می شود . به همین دلیل در ابتدا فیلتر میانه استفاده می کنم و در ادامه wiener را به کار می برم.

تصویر بعد از اعمال میانه (p8i\_filtered\_1.png)



تصویر بعد از اعمال weiner (p8i\_filtered\_2.png) با سایز ۳\*۳



تصویر  
ورودی



آخرین خروجی



## قسمت ج

کد این قسمت در p8j.m قرار دارد. ابتدا تصویر را رسم می‌کنم و بعد یک قسمت یک نواخت تصویر را انتخاب می‌کنم و در ادامه هیستوگرام آن را رسم می‌کنم:

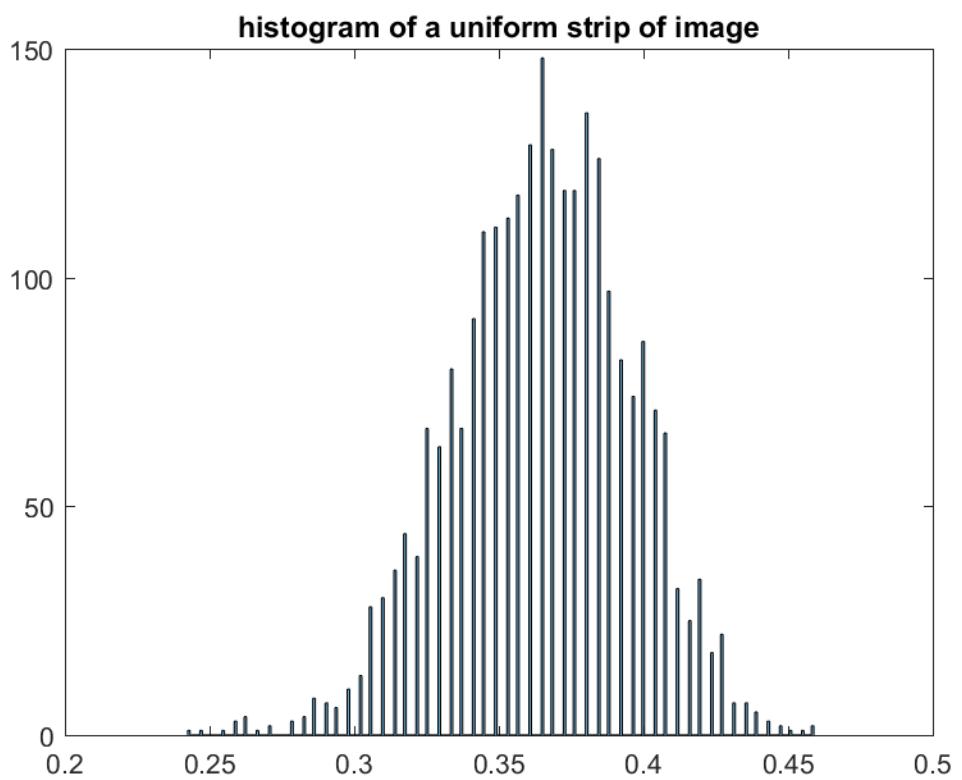
تصویر ورودی:



قسمت یک نواخت از تصویر:(p8j\_strip.png)



هیستوگرام قسمت یک نواخت (p8j\_hist\_strip\_image.png)



نویزی گووسی در هیستوگرام بالا قابل مشاهده است.

برای رفع آن از wiener با سایز  $7 \times 7$  استفاده می‌کنم، در زیر خروجی کار را مشاهده می‌کنید:



تصویر  
ورودی



آخرین خروجی



## تمرین ۹

### قسمت a

نویز را یک متغیر تصادفی  $mu(x)$  در نظر می‌گیریم، در صورتی که این متغیر تصادفی مستقل از تصویر و حالت سیستم باشد نویز از نوع **addictive** است در غیر این صورت نویز از نوع **multiplicative** است. نویز نمک فلفل یک نویز **addictive** است زیرا از یک توزیع ای استخراج می‌شود که توزیع از تصویر مستقل است.

### قسمت b

از آنجایی که نمی‌شود از فیلترهای حوزه‌ی فرکانس استفاده کرد از فیلترهای حوزه‌ی **spatial** استفاده می‌کنیم. بستگی به نویز می‌توان از انواع مختلف فیلترها استفاده کرد، به عنوان مثال در صورتی که نویز پریودی به صورت خطهای افقی باریک و روشن باشد می‌توان از **min** استفاده کرد. یا در صورتی که خطهای افقی و عمودی باریک در تصویر باشد می‌توان از انواع **mean filter** ها استفاده کرد. هر چقدر میزان نویز متناوب بیشتر باشد به طور مثال تعداد و ضخامت خطهای عمودی بیشتر باشند، از فیلترهای **min** بزرگتری استفاده می‌کنیم.

### قسمت c

یکی از روش‌های از بین بدن موج نویزی بلور کردن تصویر است، از آنجایی که از فیلترهای **spatial** نمی‌توان استفاده کرد از فیلترهای **Low Pass** استفاده می‌کنیم. همچنین می‌توان از **weiner** استفاده کرد. هر چه واریانس نویز بیشتر باشد، باید تصویر را بیشتر بلور کرد. در نتیجه از فیلتر **low pass** استفاده می‌کنیم که فرکانس‌های کمتری را از خود رد می‌کند.

### قسمت d

آن‌المنت اضافه اگر نباشد، امکان نوشتن انتقال به صورت ضرب یک ماتریس در یک وکتور نیست و باید انتقال را به صورت جمع دو وکتور نشان داد.

### قسمت e

$$\begin{matrix} \cos(-60) & \sin(-60) & 0 & x \\ -\sin(-60) & 2\cos(-60) & 20 & * \\ 1 & & & y \end{matrix}$$