

به نام خدا



درس : سیگنال‌ها و سیستم‌ها

استاد: دکتر کربلایی

گزارش پروژه شماره ۵

امین زمانی

۹۴۱۰۰۷۸۷

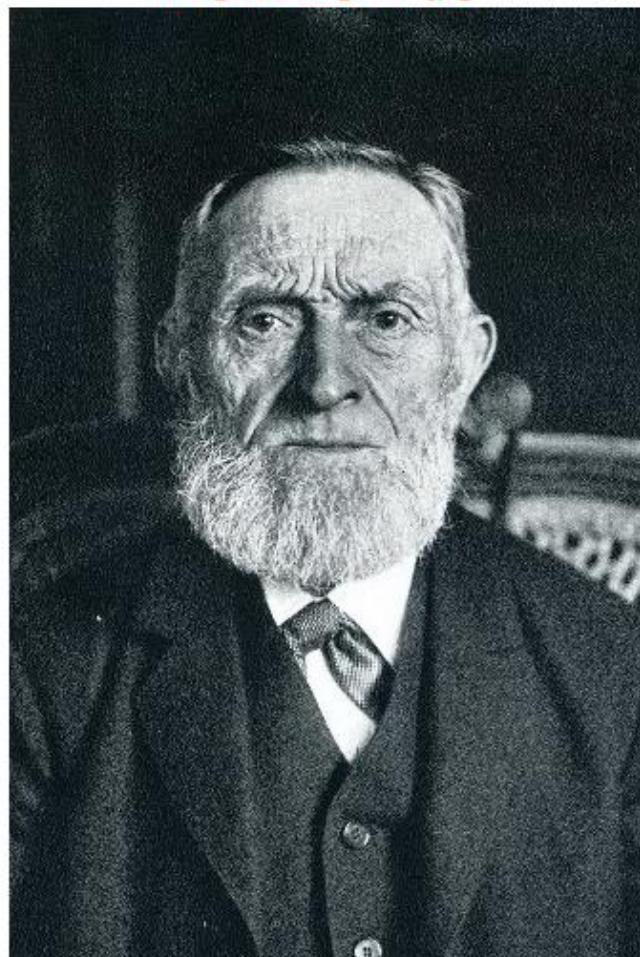
سید محمد امین منصوری

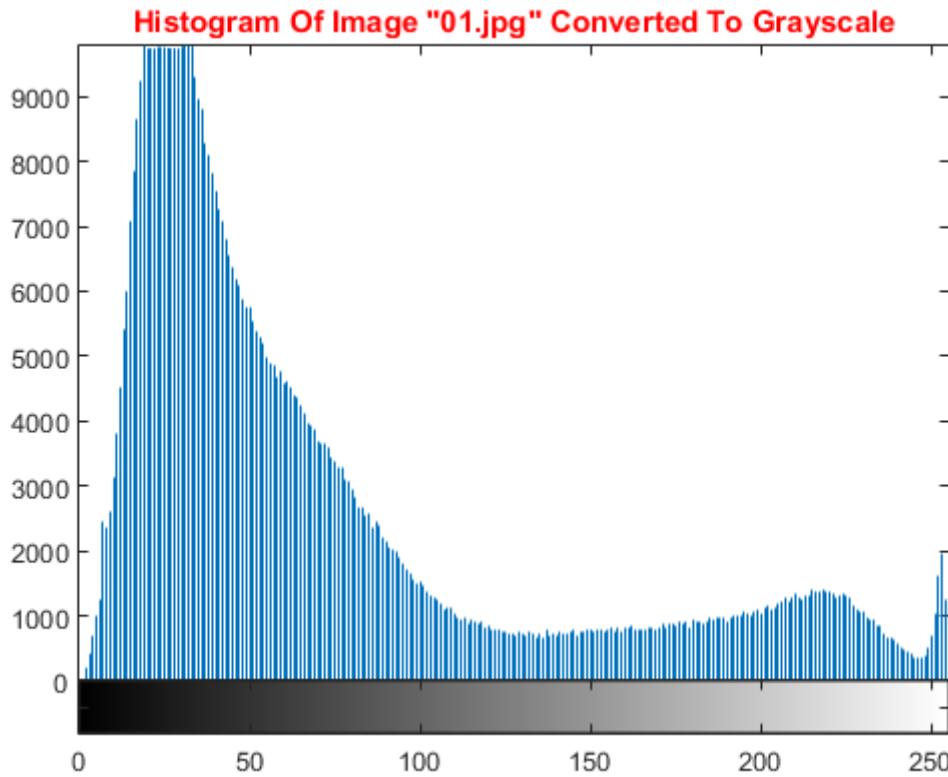
۹۴۱۰۵۱۷۴

قسمت دوم: پیش پردازش تصویر اول و آشنایی با تبدیل FFT دو بعدی

۱. توضیح کد: همان طور که مشاهده می شود با دستور `imread` تصویر در مطلب خوانده شده و در ادامه در یک پنجره با دستور `imshow` نمایش داده می شود. برای نمایش هیستوگرام نیز ابتدا با دستور `rgb2gray` همه پیکسل های تصویر در فرمت سیاه و سفید بین ۰ و ۲۵۵ مقداردهی شده و سپس با توجه به این مقادیر با دستور `imhist`، هیستوگرام مورد نظر رسم می شود. نتایج در تصاویر زیر مشاهده می شوند.

Original Image "01.jpg"





۲. ابتدا لازم است راجع به انواع نویز شناخت کوتاه و کافی بدست آوریم. در زیر توضیحی پیرامون انواع نویز تصاویر آورده می‌شود و در انتهای نتیجه‌گیری می‌کنیم.

Gaussian noise: منابع اصلی این نویز، نویز حسگرها به خاطر روش‌نایی ضعیف تصویر یا دمای بالا یا نویزهای ناشی از انتقال اطلاعات توسط مدارهای الکترونیکی است. نویز خازن‌ها و تقویت‌کننده‌ها را نیز می‌توان در این دسته قرار داد. همچنین با توجه به قضیه حد مرکزی در صورتی که اثرات مختلفی در داده‌گیری و آزمایش (در اینجا فتوکپی) اثر داشته باشند، توزیع نویز به یک توزیع گاوسی میل می‌کند.

Salt & pepper noise: در تصویر شامل این نویز در مناطق تاریک، نقاط روشن و در مناطق روشن، نقاط تاریک نامربوط به بقیه تصویر رؤیت می‌شود. ممکن است به وسیله مبدل‌های آنالوگ به دیجیتال یا خطاهای بیت در انتقال داده رخ دهد. روش حذف آن استفاده از فیلتر میانه یا درون‌یابی مقدار پیکسل‌هاست.

Shot noise: نویز غالب در قسمت‌های تاریک‌تر عکس ناشی از حسگر تصویر که معمولاً به علت نوسانات کوانتمی آماری است. نویزهای این دسته در هر پیکسل از پیکسل‌های دیگر مستقل‌اند و از توزیع پواسون پیروی می‌کنند. این توزیع مگر در شدت‌های پایین قابل تقریب زدن با توزیع گاوسی است.

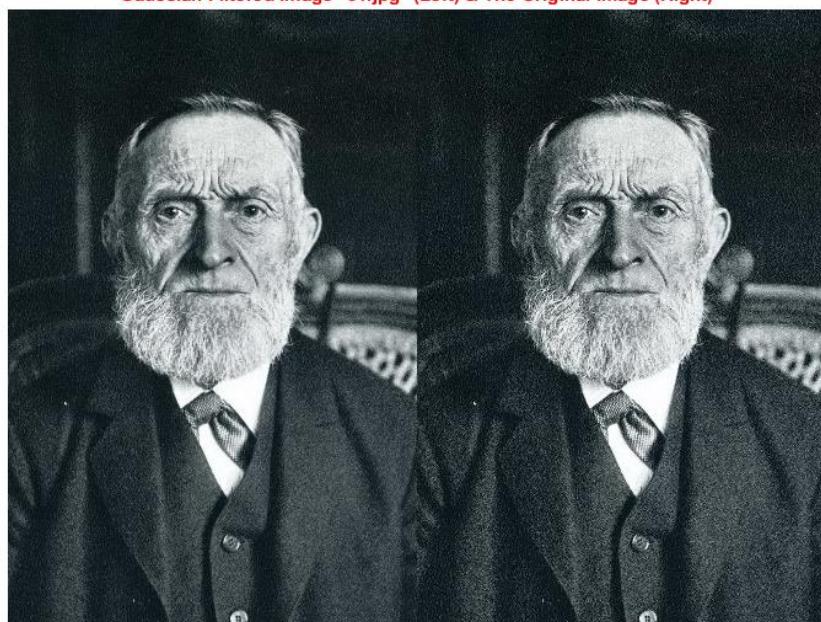
Quantization noise: ناشی از کوانتیزه کردن مقادیر پیکسل‌ها به مقادیر گسسته است. با توزیع یکنواخت تقریب زده می‌شود.

: یک نویز وابسته به سیگنال است و توزیع آماری شبیه به *Shot noise* دارد. این نویز نیز معمولاً در موارد بسیاری با توزیع گاوسی تقریب زده می‌شود.

: ناشی از تداخل الکتریکی یا الکترومکانیکی هنگام گرفتن عکس است. تصویر با این نویز این‌گونه به نظر می‌رسد که یک الگوی تکرارشونده روی تصویر اضافه شده است.

نتیجه‌گیری و توضیح کد: به نظر می‌رسد احتمال وجود نویز گاوسی از بقیه بیشتر باشد. البته در مناطقی دانه دانه بودن تصویر دیده می‌شود بنابراین احتمال وجود نویز از نوع *Salt & pepper* نیز وجود دارد بنابراین برای بررسی دقیق‌تر از ۳ نوع فیلتر استفاده می‌کنیم. ابتدا با دستور *fspecial* و ورودی‌های *average* و *gaussian* برای ساختن ماتریس‌های لازم از قبل تعریف شده، آنها را برای عمل کردن روی تصویر می‌سازیم. فیلتر اول میانگین اطراف هر پیکسل و فیلتر دوم یک نمودار دو بعدی گاوسی را بر روی هر پیکسل و اطراف آن اعمال کرده و نتیجه را به جای پیکسل مرکزی می‌گذارد. در ادامه به وسیله دستور *imfilter* و ماتریس‌های فیلتر فوق تصویر *img1* را فیلتر می‌کنیم. فیلتر آخر برای حذف کردن نویز *Salt & pepper* است. همان‌طور که گفته شد باید با فیلتر دو بعدی میانه آن را حذف کرد. برای دادن یک تصویر به این فیلتر (*medfilt2*) باید ماتریس این تصویر دو بعدی باشد لذا از دستور *rgb2gray* استفاده می‌کنیم و سپس این تصویر جدید را به ورودی فیلتر می‌دهیم. نهایتاً همه فیلترها با هم نیز اعمال شده و در کنار تصویر اصلی نشان داده می‌شود. (*img1FilteredAll*) در ادامه، هر تصویر فیلتر شده را در کنار تصویر اصلی برای مقایسه عملکرد فیلتر و مناسب بودن آن نشان می‌دهیم. (با دستور *imshowpair* و صفت *montage* پس از دادن دو تصویر مورد نظر) این نتایج در تصاویر زیر مشاهده می‌شوند.

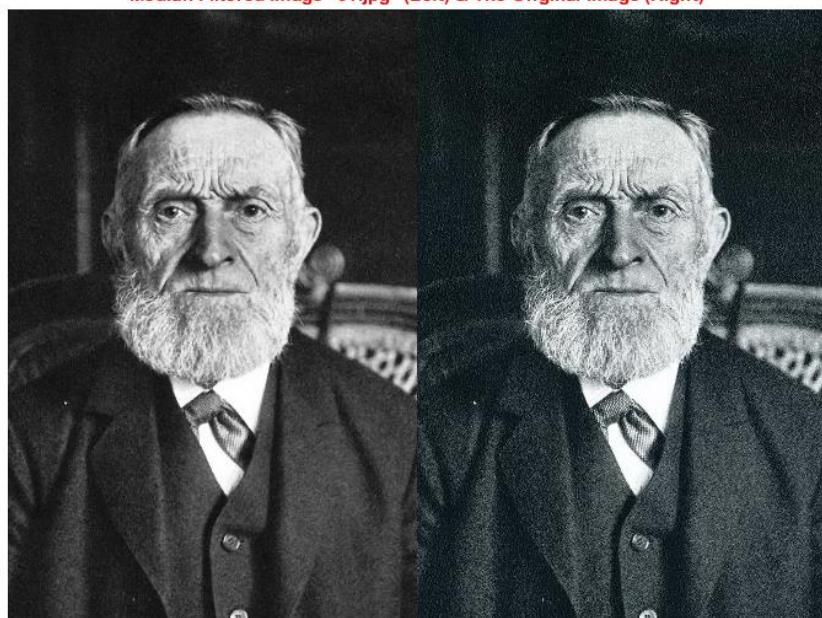
Gaussian Filtered Image "01.jpg" (Left) & The Original Image (Right)



Average Filtered Image "01.jpg" (Left) & The Original Image (Right)



Median Filtered Image "01.jpg" (Left) & The Original Image (Right)



All Filters Used For Image "01.jpg" (Left) & The Original Image (Right)



فیلتر گاووسی برخلاف انتظار عملکرد مورد انتظار را ندارد و تنها تا حدی از دانه دانه بودن تصویر می‌کاهد. فیلتر میانگین همانطور که مانند فیلتر *Moving Average* در حالت یک بعدی (مثلاً برای صوت تمرين ۱ و ۴) انتظار داریم، تصویر را هموار ساخته و کیفیت آن را بالاتر می‌برد. فیلتر میانه به نظر می‌رسد از آن هم بهتر عمل می‌کند و در رنگ تصویر نیز مؤثر است. تصویر هموار شده و دانه بودن تا حد مطلوبی کاهش یافته‌است. در انتها نیز ترکیب فیلترها و نتیجه نهایی مشاهده می‌شود که ویژگی‌های هر ۳ فیلتر فوق در آن مشاهده می‌شود. به نظر می‌رسد اثر نویز *Salt & pepper* غالب است و با دقت در مناطق تاریک و حضور نقطه‌های سفید بسیار می‌توان به آن پی‌برد.

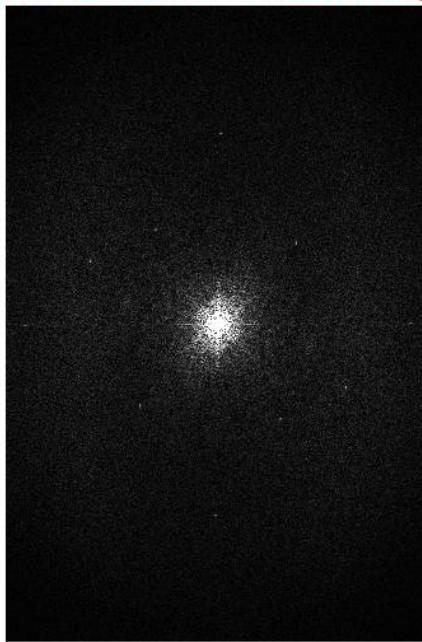
۳. برای نمایش تبدیل فوریه راههای مختلفی وجود دارد که ما به وسیله ۲ تا از آن‌ها اندازه و فاز تبدیل فوریه‌های مورد نظر را رسم می‌کنیم.

روش اول: تابع *FFT2* تابعی است که ما خودمان تعریف کردیم و با گرفتن تصویر، ابتدا آن را به مقیاس *grayscale* برد و سپس مقادیر آن را به فرمت *double* می‌برد تا برای ورودی تبدیل فوریه دو بعدی مناسب شود. این ماتریس تصویر جدید به ورودی *fft2* داده می‌شود که تبدیل دو بعدی آن را می‌گیرد. همانند تمرين‌های قبلی برای متقارن شدن نتیجه تبدیل فوریه از دستور *fftshift* استفاده می‌کنیم. این ماتریس جدید خروجی تابع ما خواهد بود. پس از آن این ماتریس تصویر جدید که مربوط به تصویر تبدیل فوریه است می‌باشد به وسیله دستور *imshow* نمایش داده شود. اما برای اینکه محدوده فرکانسی مورد نظر ما نمایش داده شود در قسمت دوم ورودی این تابع محدوده مورد نظر و مناسب را وارد می‌کنیم. با آزمون و خطابه سادگی می‌توان محدوده مناسبی که در آن تبدیل فوریه تصویر وجود دارد را بدست آورد. (در صورت

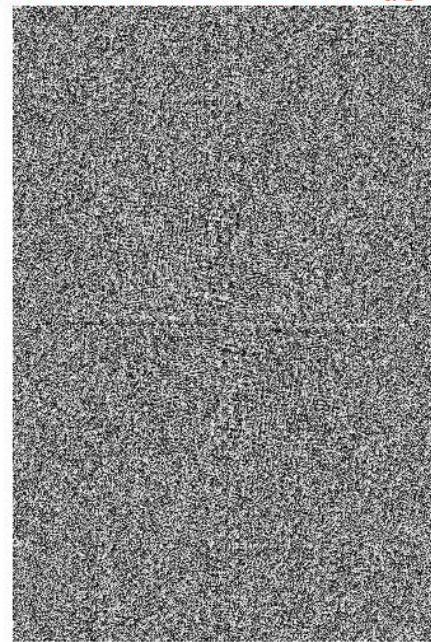
مشخص کردن محدوده نامناسب یا همه تصویر سیاه خواهد بود و فقط نقطه‌ای روشن در مرکز که اطلاعات مورد نظر ماست دیده می‌شود و یا همه تصویر روشن با نقاطی تاریک مشاهده می‌شود که نشان می‌دهد در قسمت مورد نظر بزرگ‌نمایی کرده‌ایم.) برای فاز هم که مشخصاً نتیجه در بازه $[\pi, -\pi]$ برای ما مهم است.

روش دوم: ابتدا تبدیل فوریه دو بعدی گرفته می‌شود. برای متقارن شدن از *fftshift* استفاده می‌کنیم. برای اندازه تبدیل فوریه از قدر مطلق مقدار اخیر استفاده می‌کنیم. برای فاز نیز از *angle* استفاده می‌کنیم. در ادامه برای نمایش بهتر مقادیر زیاد و کوچک در گستره مناسب از لگاریتم اندازه تبدیل فوریه استفاده می‌کنیم. (به علاوه یک شدن برای جلوگیری از محاسبه $\log(0)$ است). برای مقیاس کردن بین صفر(سیاه) و یک(سفید) نیز از دستور *mat2gray* استفاده می‌کنیم. فاز نیز بدون تغییر نمایش داده می‌شود. این نتایج در تصاویر زیر مشاهده می‌شوند. برای محدوده نمایش نیز از مقدار دیفالت مطلب استفاده می‌کنیم.

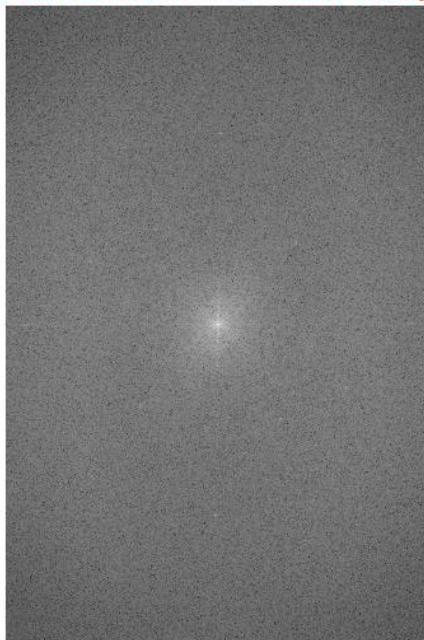
Method 1 : Absolute Value Of Fourier Transform Of "01.jpg" Image



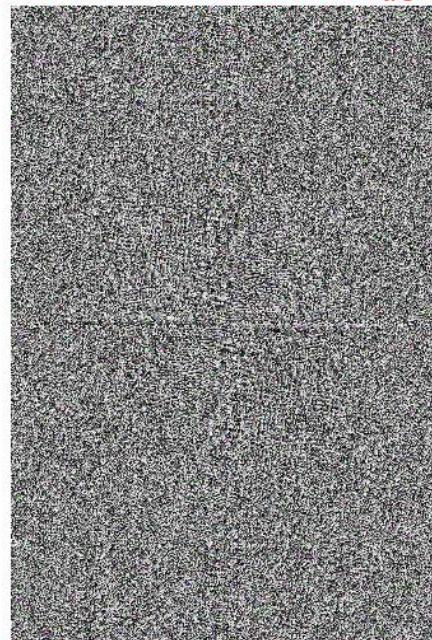
Method 1 : Phase Of Fourier Transform Of "01.jpg" Image



Method 2 : Absolute Value Of Fourier Transform Of "01.jpg" Im

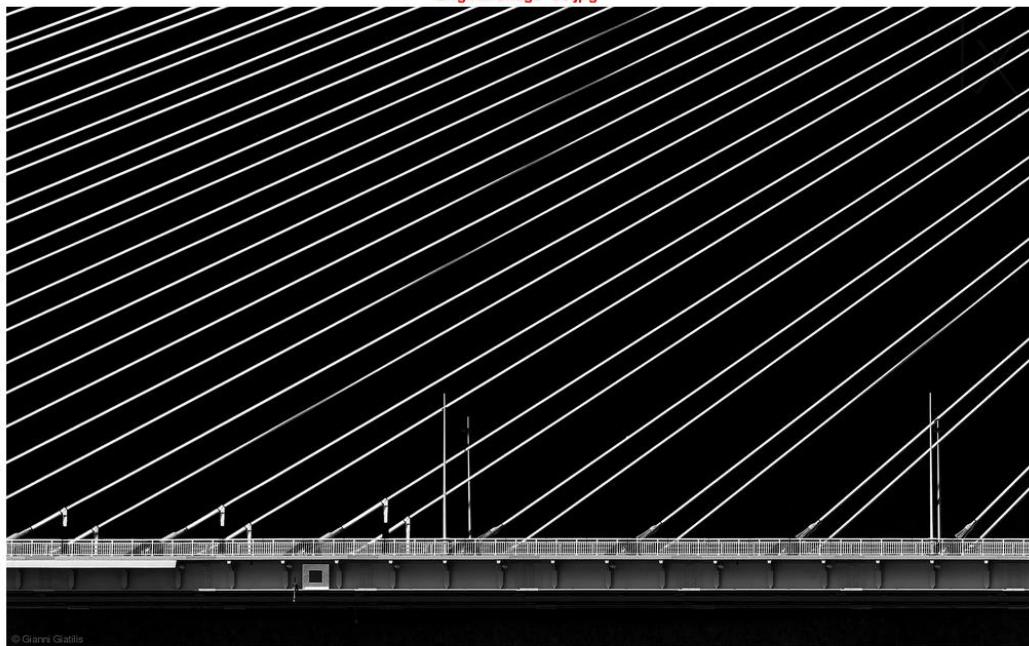


Method 2 : Phase Of Fourier Transform Of "01.jpg" Image

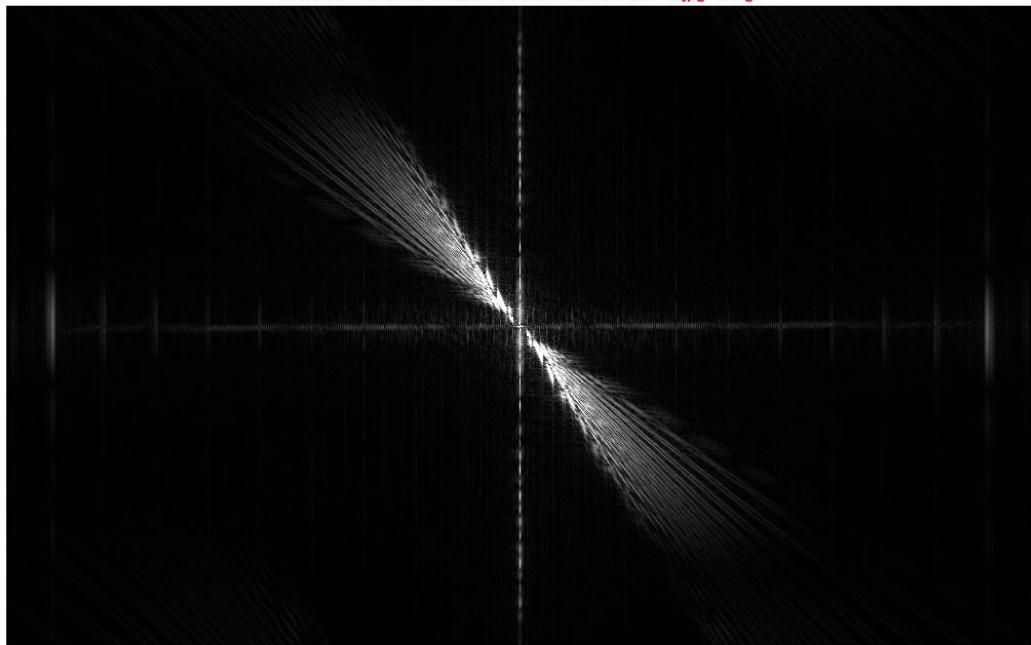


۴. این تصاویر نیز در زیر مشاهده می‌شوند.

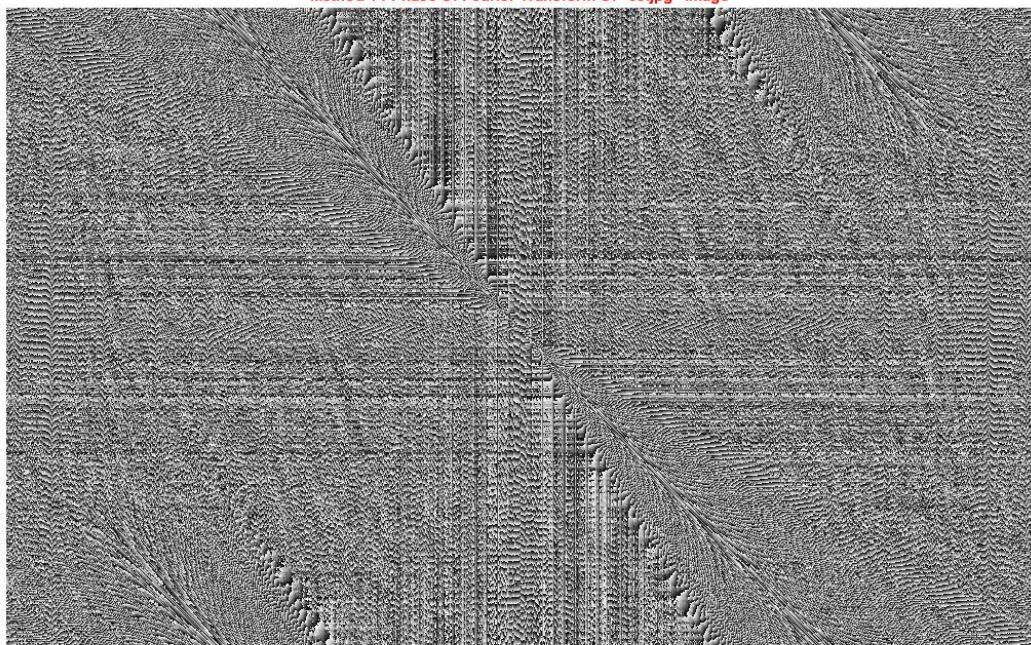
Original Image "03.jpg"



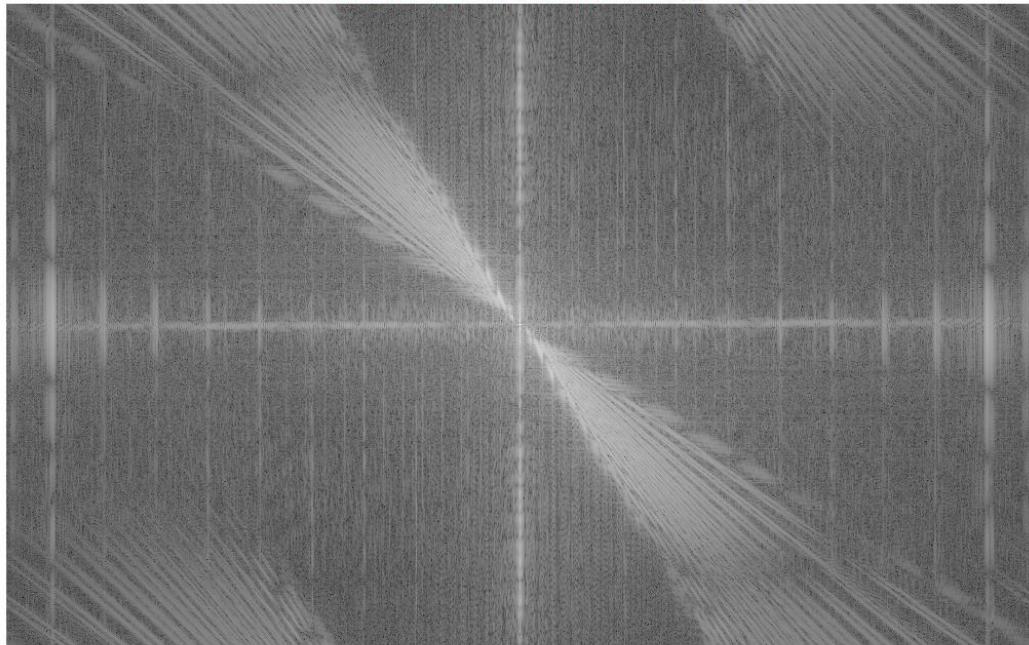
Method 1 : Absolute Value Of Fourier Transform Of "03.jpg" Image



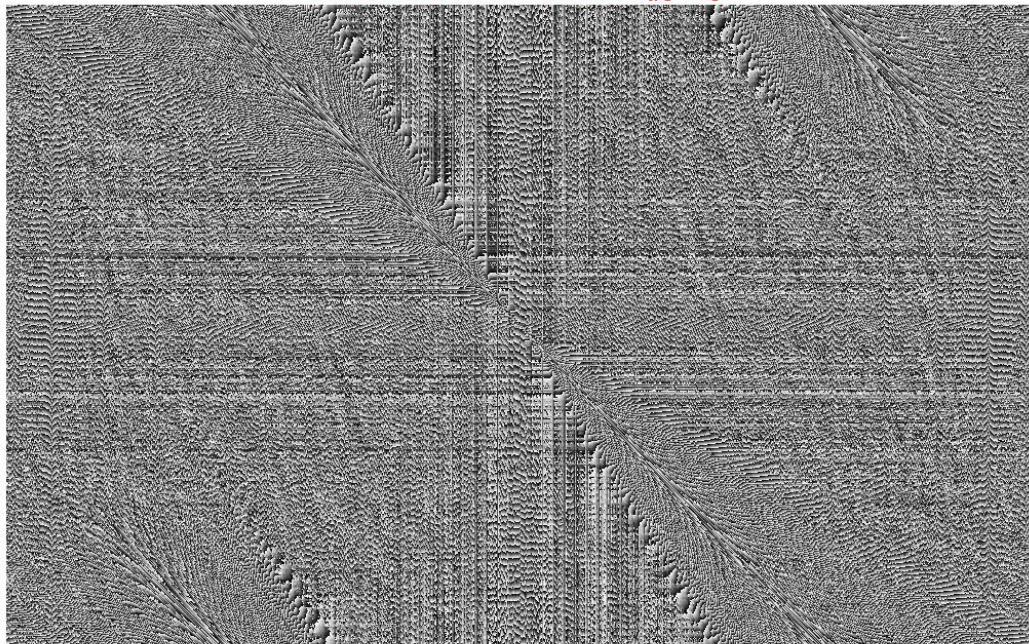
Method 1 : Phase Of Fourier Transform Of "03.jpg" Image



Method 2 : Absolute Value Of Fourier Transform Of "03.jpg" Image



Method 2 : Phase Of Fourier Transform Of "03.jpg" Image



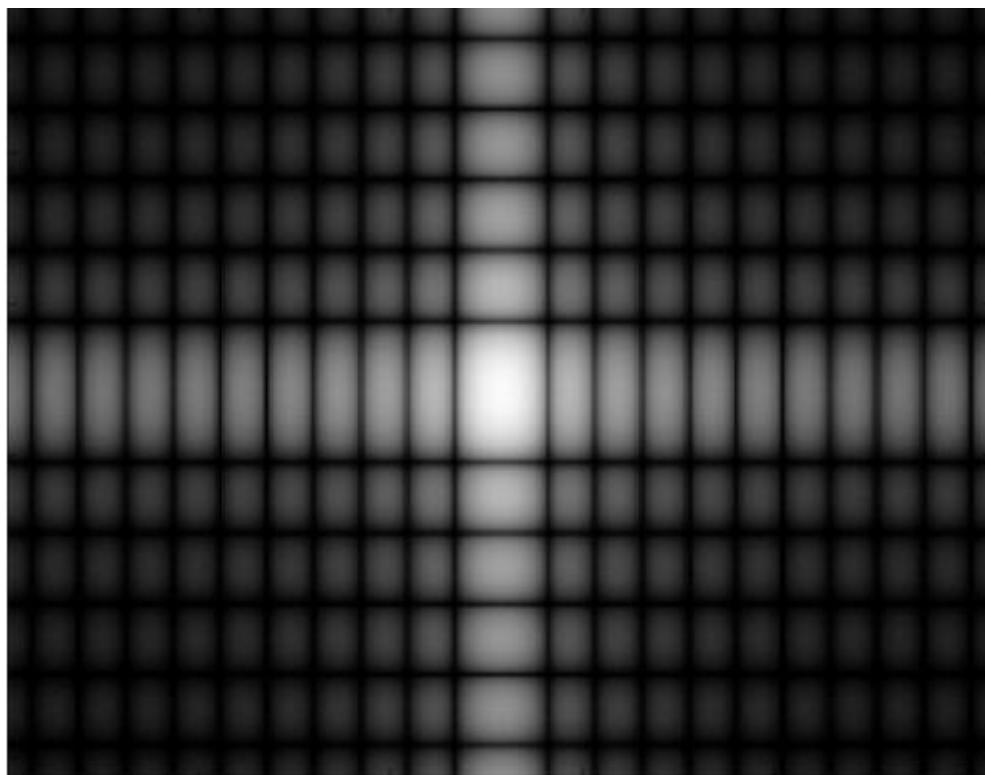
همان‌طور که در تصویر ۳ ملاحظه می‌کنیم، چند روند مهم به چشم می‌خورد. مهم‌ترین آن‌ها خط‌های مورب مربوط به سیم‌های پل هستند که همگی تا حد خوبی موازی اند و اگر در جهت عمود بر آن‌ها حرکت کنیم نوساناتی در آن راستا خواهیم دید و این به معنی وجود هارمونیک‌ها در این جهت است. همان‌طور که تصاویر اندازه تبدیل فوریه نشان می‌دهند، در راستای عمود بر این سیم‌ها هارمونیک داریم و چون جهت سیم‌ها اندکی تفاوت دارد در راستای عمود بر هر کدام، این هارمونیک‌ها مشاهده می‌شوند. همچنین در راستای عمودی هم وجود هارمونیک‌ها مشخص است و این به علت وجود نرده‌های روی پل و تیرهای نگه دارنده است که باعث

می‌شوند در حرکت افقی نوسان داشته باشیم و این هارمونیک‌ها را تولید کنند. در تصویر با روش دوم در راستای افقی نیز وجود هارمونیک‌ها مشاهده می‌شود که ناشی از خطوط افقی تصویر است. این خطوط شامل بالا و پایین نرده‌ها و همچنین لبه‌های بالایی و پایینی پل است. همچنین فاز نیز در جهت سیم‌ها نوار نوار است.

قسمت سوم: جداسازی فرکانسی تصویر اول و یافتن مرزها

۱. توضیح کد: در حلقه f برای 4 مقدار مختلف k عملیات زیر انجام می‌شود.

ابتدا با دستور $fspecial$ ماتریس فیلتر میانگین‌گیر ساخته می‌شود و سپس تصویر با آن فیلتر می‌شود. پس از آن این تصویر نمایش داده می‌شود. پس از آن نوبت به تبدیل فوریه می‌رسد. باز هم با هر دو روش اندازه تبدیل فوریه را نمایش می‌دهیم. نهايتا نیز هيستوگرام هر تصویر پس از فیلتر شدن با فیلتر MA دو بعدی نمایش داده می‌شود. نتایج در تصاویر زیر مشاهده می‌شوند. مشاهده می‌شود با زیاد کردن اندازه پنجره، هر نقطه به میانگین کل نزدیک‌تر شده و تصویر تارتر می‌شود. علت مربع مرتع شدن تصویر تبدیلات فوریه، ضرب شدن تبدیل فوریه جعبه فیلتر $moving\ average$ در تبدیل فوریه تصویر اصلی است. می‌دانیم که تبدیل فوریه یک جعبه مانند تصویر زیر است و عمل فیلتر کردن نیز نوعی کانولوشن. بنابراین تصاویر مشاهده شده با این الگو قابل انتظارند.(با تغییر طول پنجره‌ها این تصاویر در تبدیل فوریه نیز عوض می‌شوند).



* در تصاویر اندازه تبدیل فوریه که با دو روش و در یک *subplot* رسم شده‌اند، عنوان تصویر سمت راست تمام این *subplot* ها به جای *img01* باید *img03* باشد و ما متاسفانه به دلیل وقت‌گیر بودن اصلاح دوباره آنرا اصلاح نکردیم.

Average Filtered Image "01.jpg" with k = 12



Average Filtered Image "01.jpg" with k = 8



Average Filtered Image "01.jpg" with k = 20



Average Filtered Image "01.jpg" with k = 16



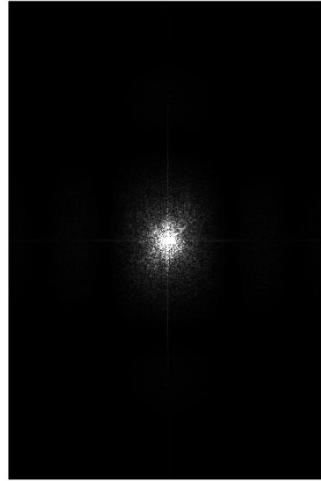
Average Filtered Image "01.jpg" with k = 24



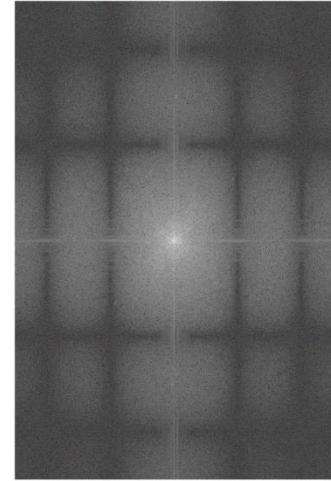
Average Filtered Image "01.jpg" with k = 28



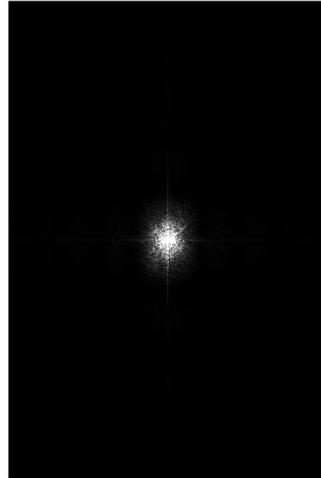
Method 1 : Absolute Value Of Fourier Transform Of Average Filtered Image "01.jpg" with k = 2



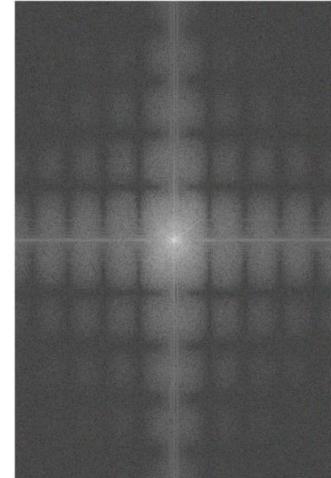
Method 2 : Absolute Value Of Fourier Transform Of "03.jpg" Image with k = 2



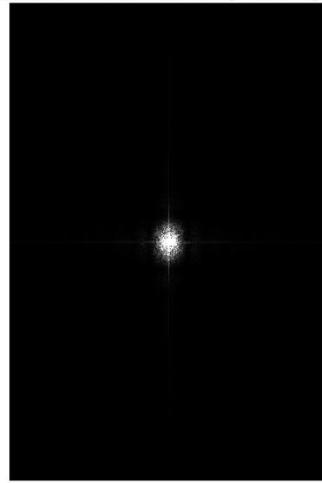
Method 1 : Absolute Value Of Fourier Transform Of Average Filtered Image "01.jpg" with k = 4



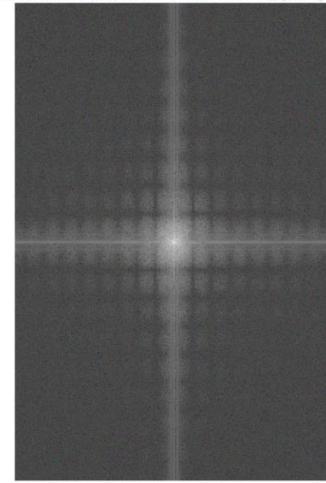
Method 2 : Absolute Value Of Fourier Transform Of "03.jpg" Image with k = 4



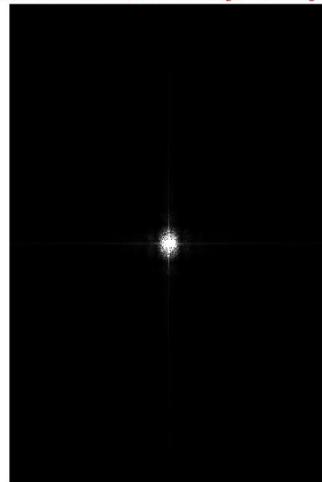
Method 1 : Absolute Value Of Fourier Transform Of Average Filtered Image "01.jpg" with k = 8



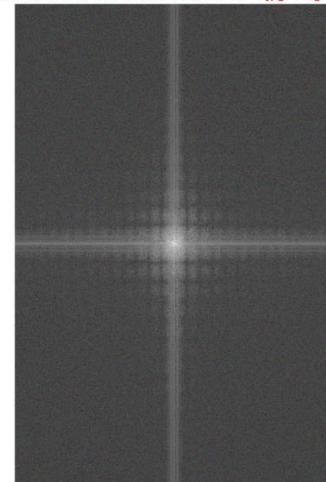
Method 2 : Absolute Value Of Fourier Transform Of "03.jpg" Image with k = 8



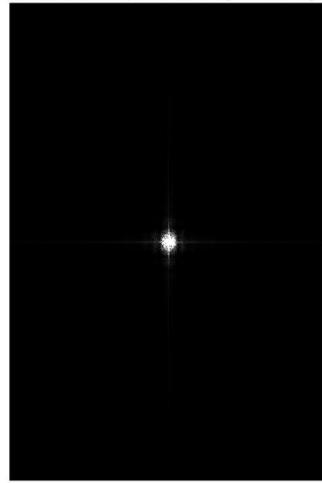
Method 1 : Absolute Value Of Fourier Transform Of Average Filtered Image "01.jpg" with k = 12



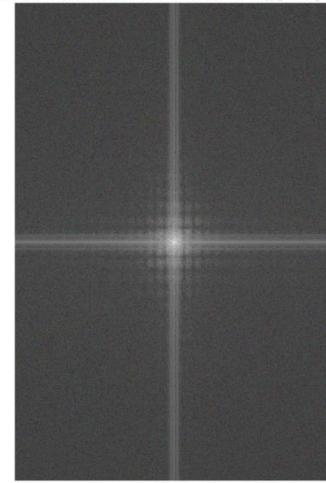
Method 2 : Absolute Value Of Fourier Transform Of "03.jpg" Image with k = 12



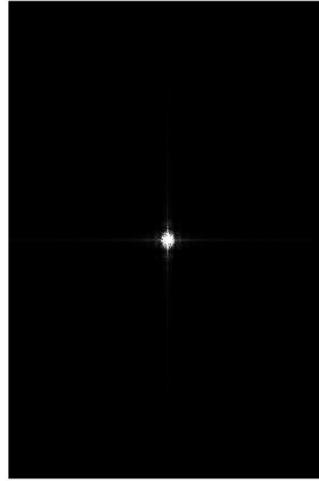
Method 1 : Absolute Value Of Fourier Transform Of Average Filtered Image "01.jpg" with k = 16



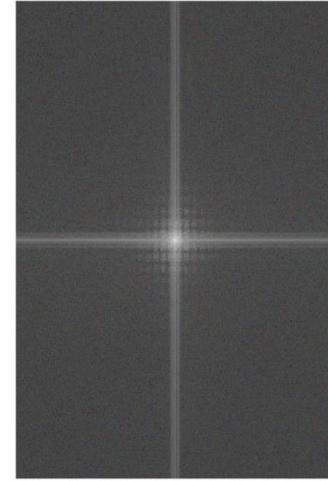
Method 2 : Absolute Value Of Fourier Transform Of "03.jpg" Image with k = 16



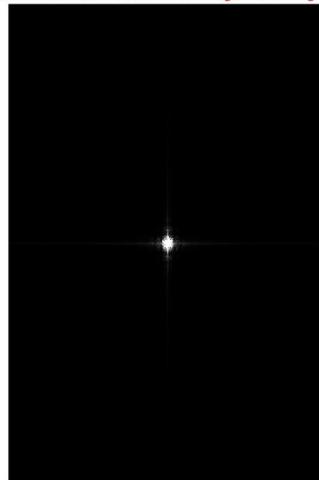
Method 1 : Absolute Value Of Fourier Transform Of Average Filtered Image "01.jpg" with k = 20



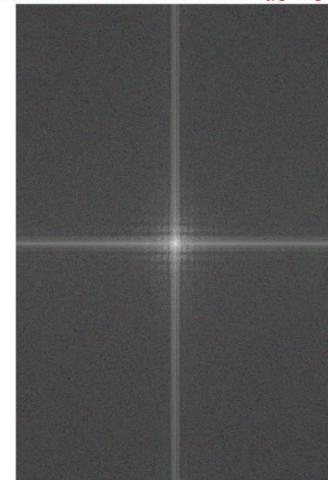
Method 2 : Absolute Value Of Fourier Transform Of "03.jpg" Image with k = 20



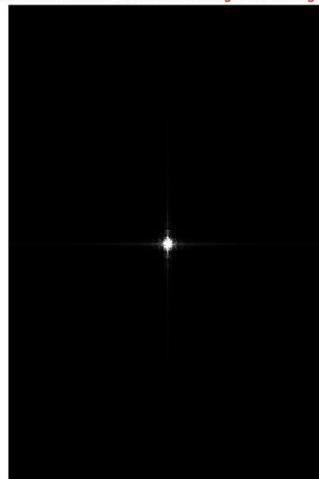
Method 1 : Absolute Value Of Fourier Transform Of Average Filtered Image "01.jpg" with k = 24



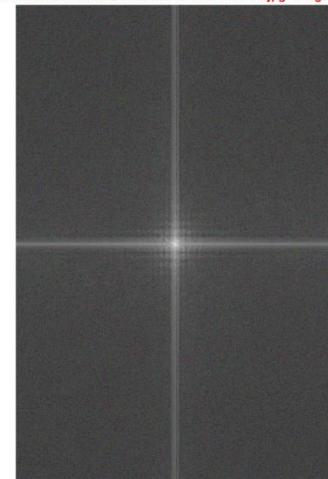
Method 2 : Absolute Value Of Fourier Transform Of "03.jpg" Image with k = 24

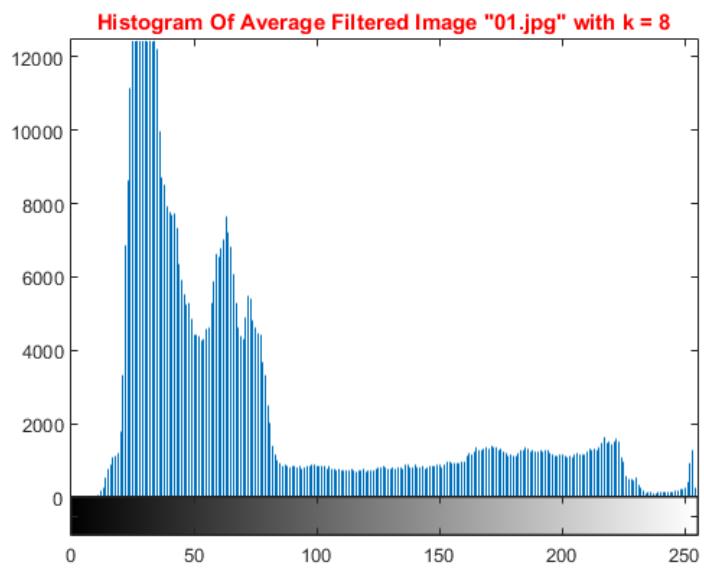
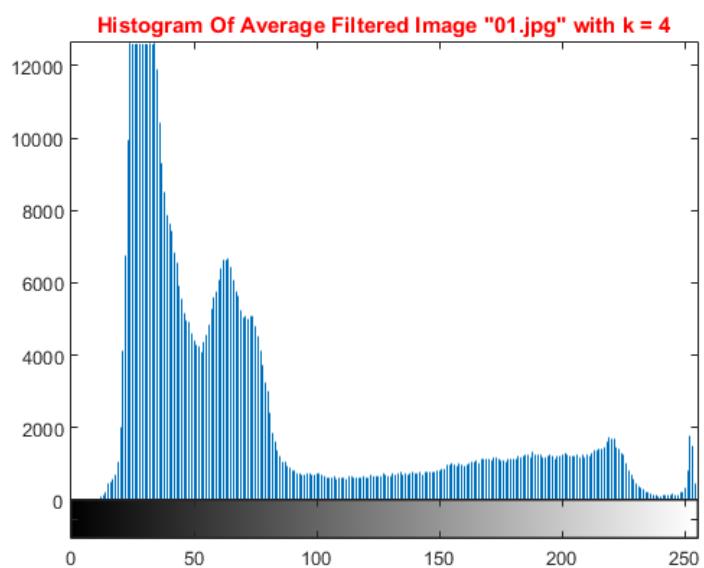
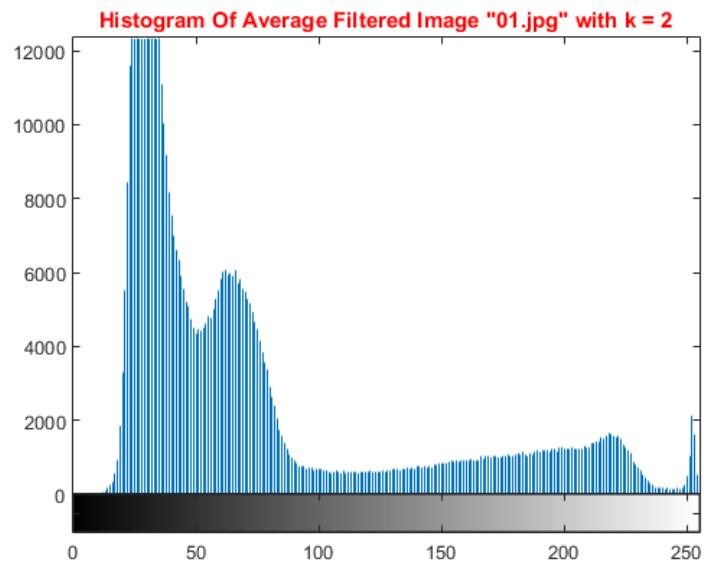


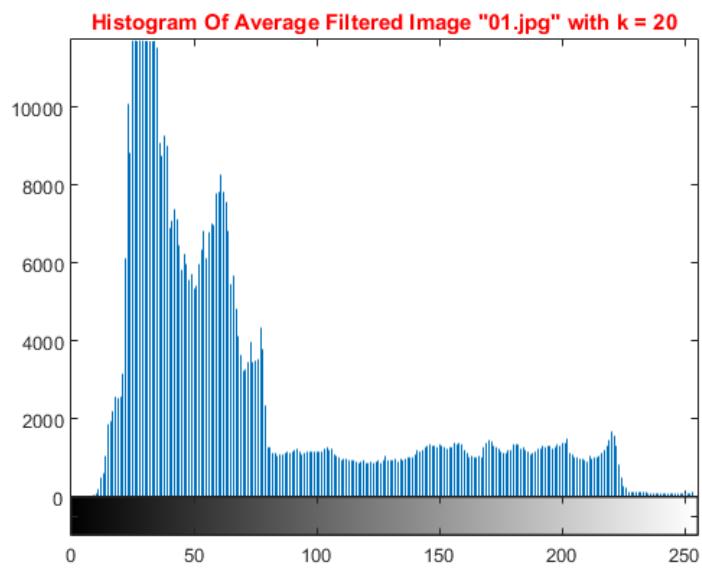
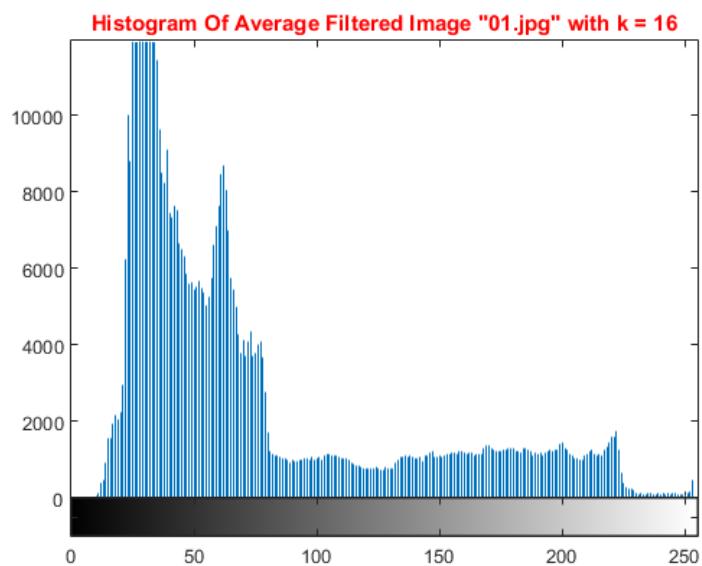
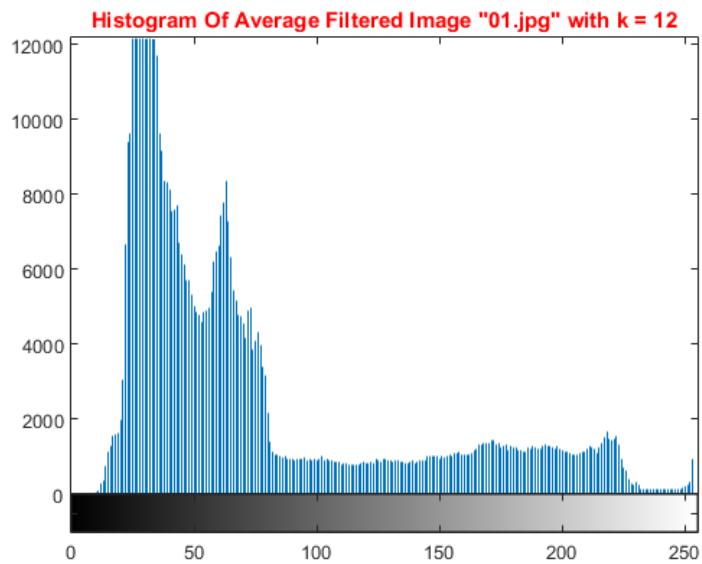
Method 1 : Absolute Value Of Fourier Transform Of Average Filtered Image "01.jpg" with k = 28

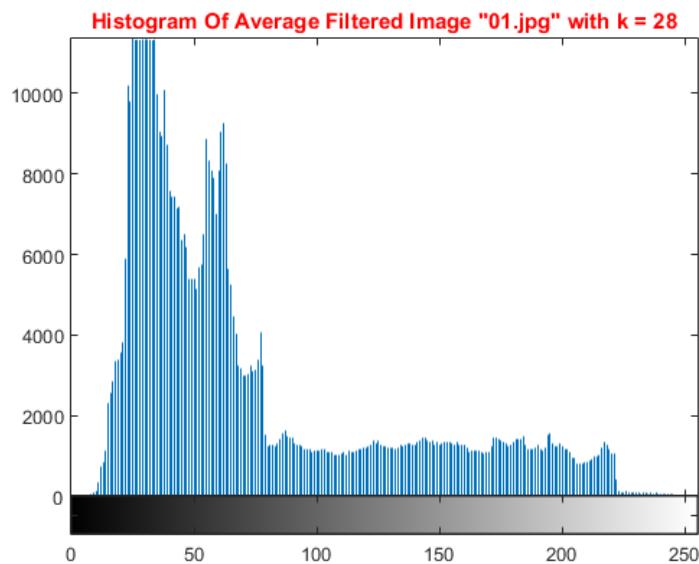
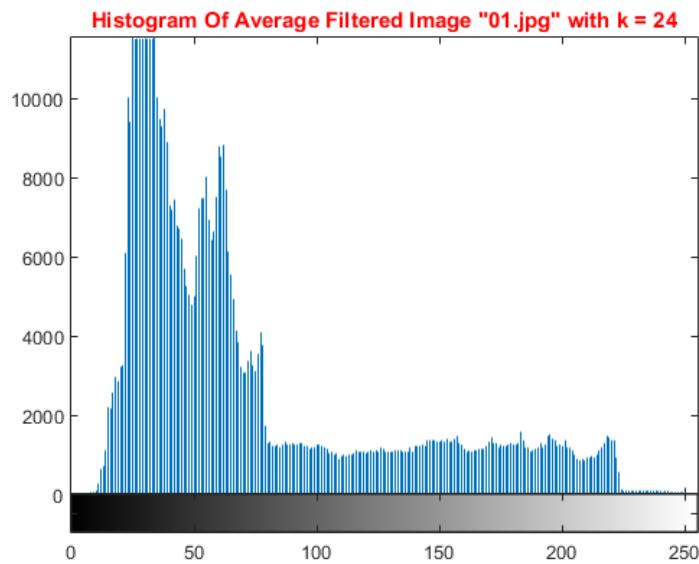


Method 2 : Absolute Value Of Fourier Transform Of "03.jpg" Image with k = 28



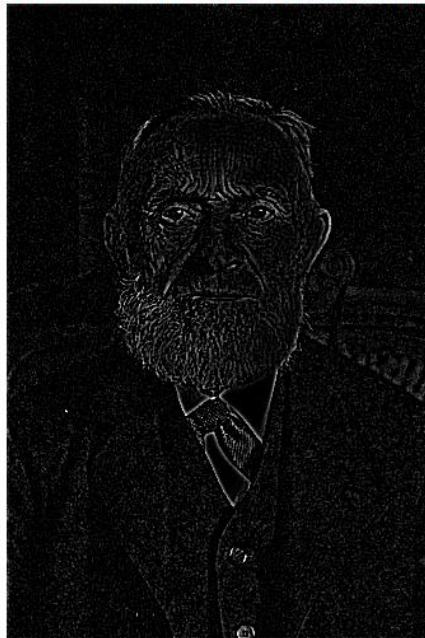




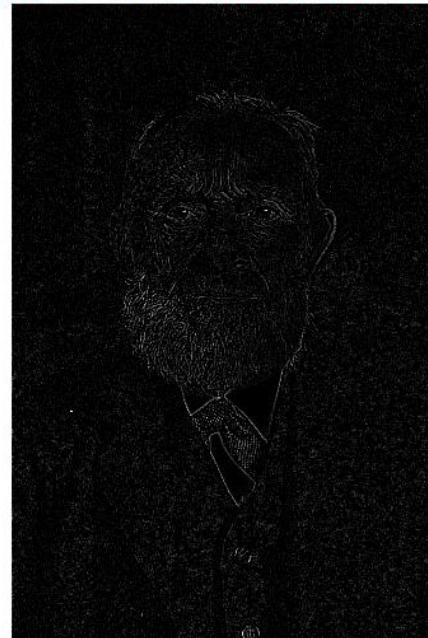


۲. توضیح کد: ابتدا تصاویر با همان فیلتر میانگین گرفته می‌شوند. سپس باید از تصویر اصلی کم شوند. اما نتیجه مطلوب برای مرزاها بدست نیامد و تصویر به مقدار زیادی دانه بود بنابراین تصمیم گرفتیم قبل از کم کردن از تصویر اصلی، ابتدا تصویر را از فیلتر میانه عبور دهیم و سپس تصویر تار شده را از آن کم کنیم. به این ترتیب به نتیجه مورد انتظار دست یافتیم. برای کم کردن از تصویر اصلی و هم بعد شدن تصویر میانگین گرفته شده با تصویر اصلی از `rgb2gray` استفاده کردیم. برای نمایش نتیجه نیز، مقدار پیکسل‌ها را در ۳ ضرب کردیم تا شدت مرزاها افزایش یابد و واضح‌تر دیده شوند. نتایج در زیر قابل مشاهده‌اند.

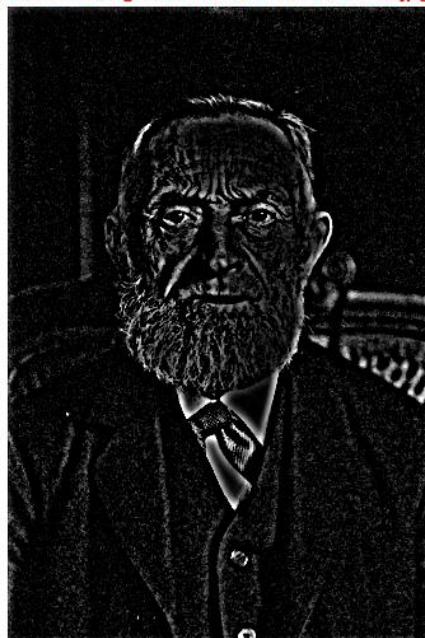
Difference Between Original & Blurred Versions of "01.jpg" for k = 4



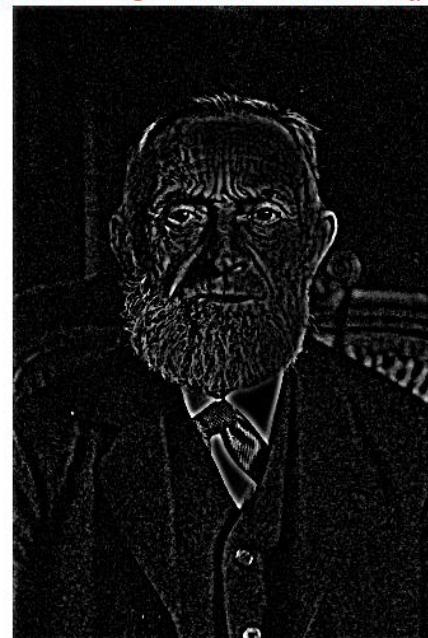
Difference Between Original & Blurred Versions of "01.jpg" for k = 2



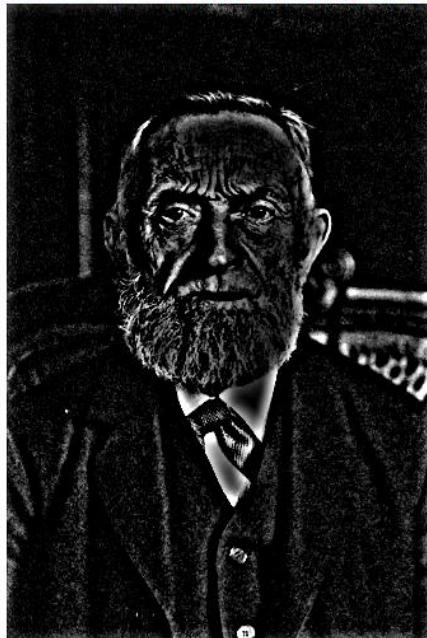
Difference Between Original & Blurred Versions of "01.jpg" for k = 12



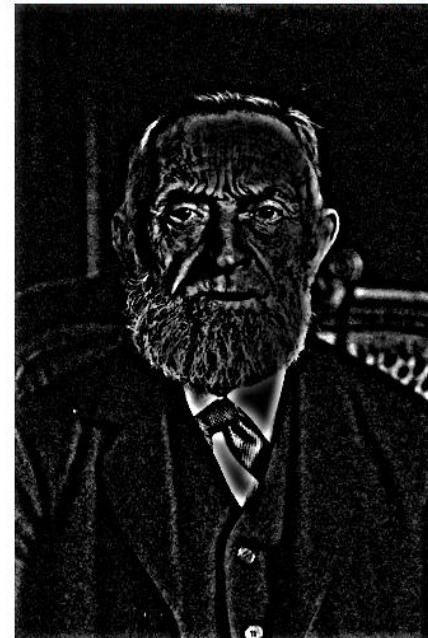
Difference Between Original & Blurred Versions of "01.jpg" for k = 8



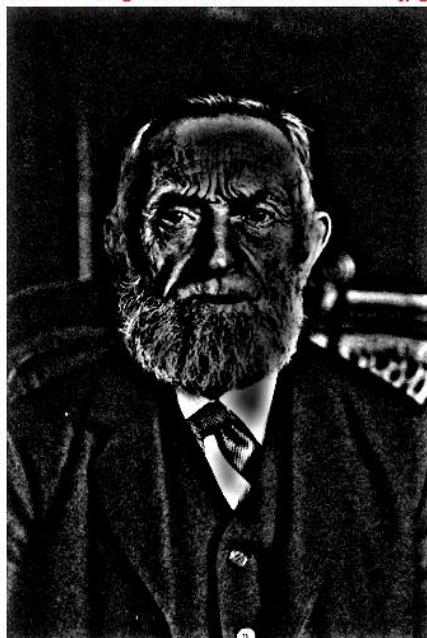
Difference Between Original & Blured Versions of "01.jpg" for k = 20



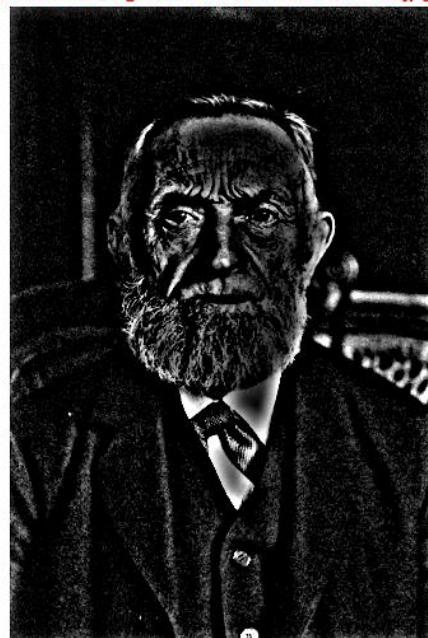
Difference Between Original & Blured Versions of "01.jpg" for k = 16



Difference Between Original & Blured Versions of "01.jpg" for k = 28

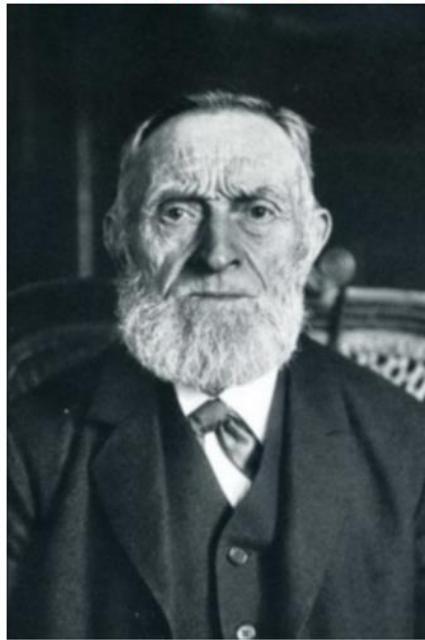


Difference Between Original & Blured Versions of "01.jpg" for k = 24

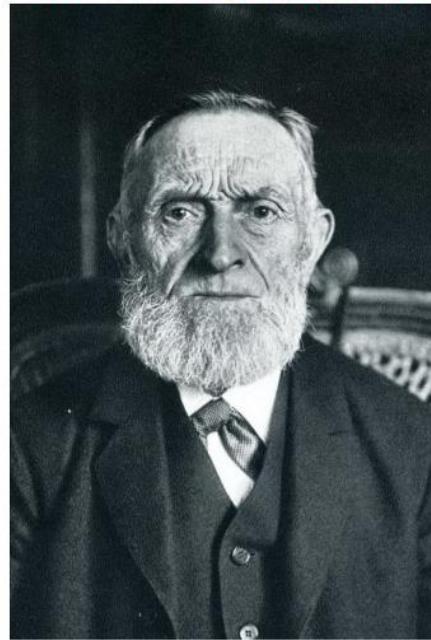


۳. این بار برای تعداد k کمتری این عملیات را انجام می‌دهیم. نتایج همه عملیات‌های سوال ۱ و ۲ را در زیر آورده‌ایم.

Gaussian Filtered Image "01.jpg" with k = 10



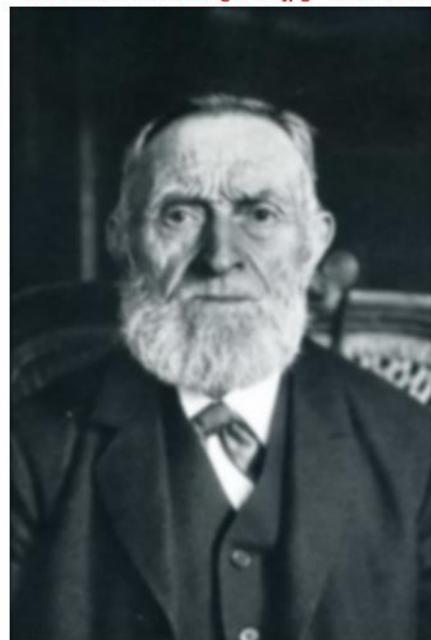
Gaussian Filtered Image "01.jpg" with k = 4



Gaussian Filtered Image "01.jpg" with k = 22



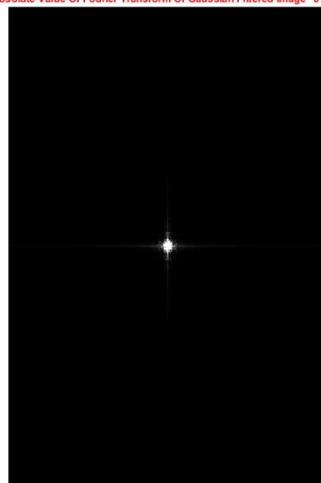
Gaussian Filtered Image "01.jpg" with k = 16



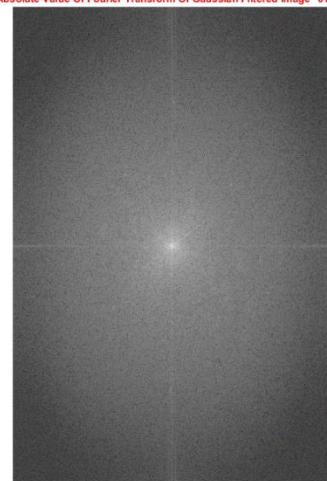
Gaussian Filtered Image "01.jpg" with k = 28



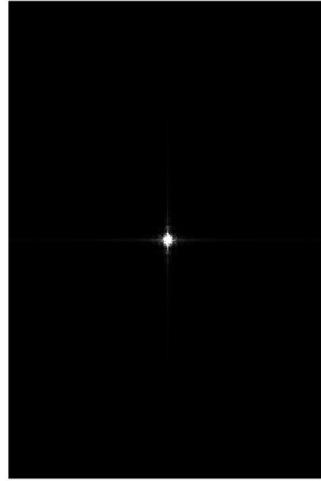
Method 1 : Absolute Value Of Fourier Transform Of Gaussian Filtered Image "01.jpg" with k = 4



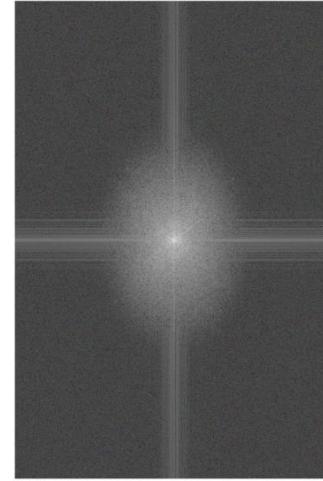
Method 2 : Absolute Value Of Fourier Transform Of Gaussian Filtered Image "01.jpg" with k = 4



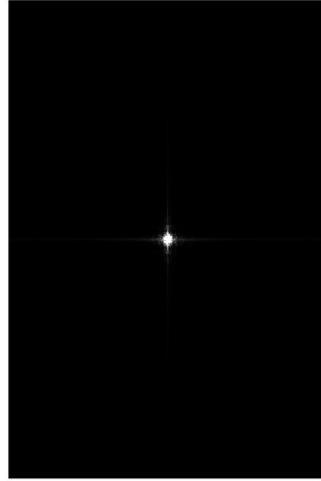
Method 1 : Absolute Value Of Fourier Transform Of Gaussian Filtered Image "01.jpg" with k = 10



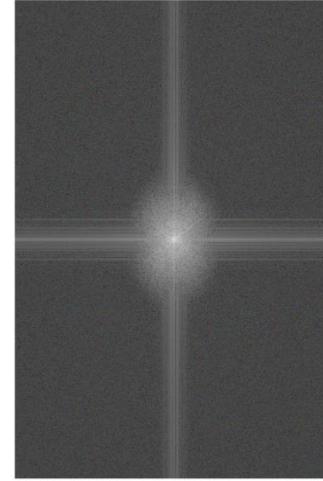
Method 2 : Absolute Value Of Fourier Transform Of Gaussian Filtered Image "01.jpg" with k = 10



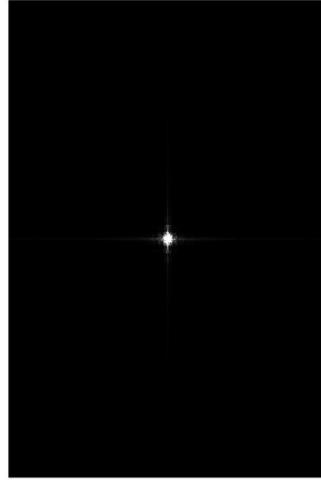
Method 1 : Absolute Value Of Fourier Transform Of Gaussian Filtered Image "01.jpg" with k = 16



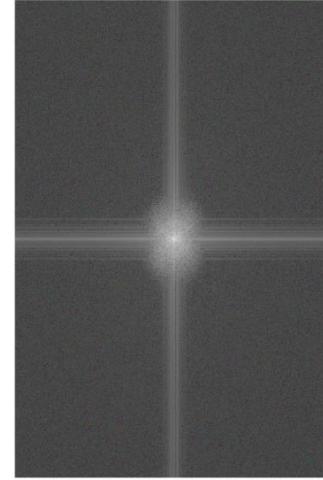
Method 2 : Absolute Value Of Fourier Transform Of Gaussian Filtered Image "01.jpg" with k = 16



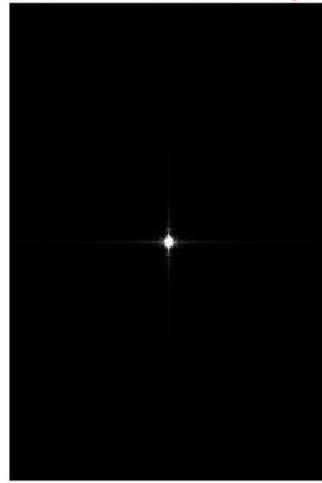
Method 1 : Absolute Value Of Fourier Transform Of Gaussian Filtered Image "01.jpg" with k = 22



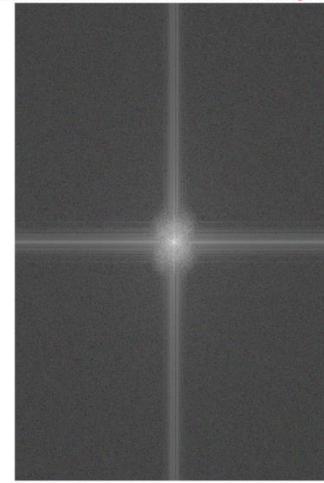
Method 2 : Absolute Value Of Fourier Transform Of Gaussian Filtered Image "01.jpg" with k = 22



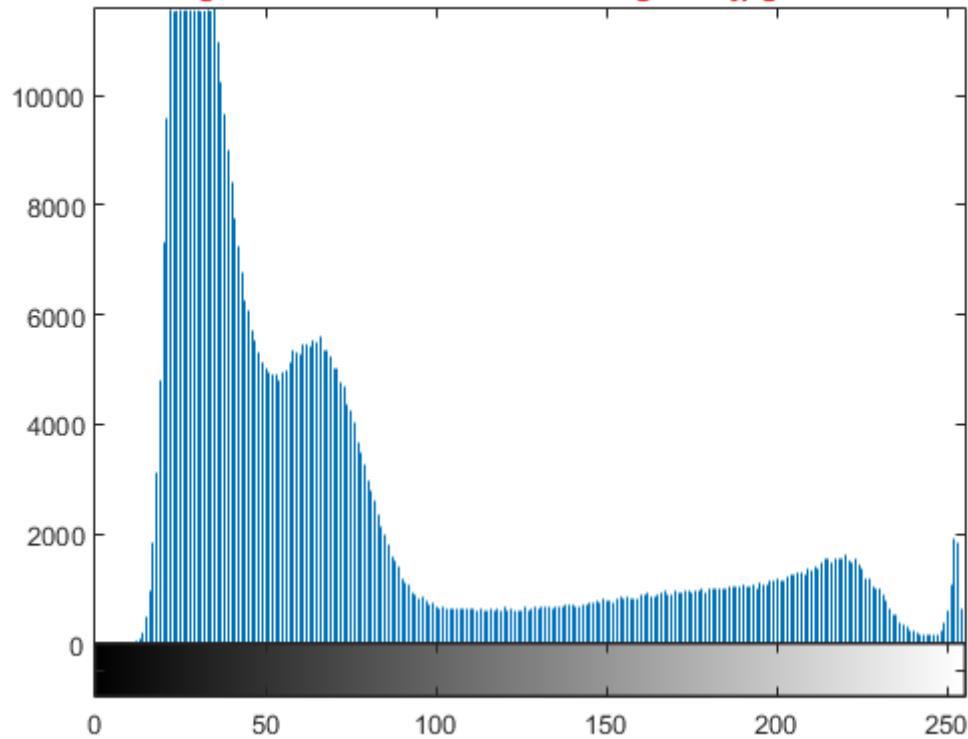
Method 1 : Absolute Value Of Fourier Transform Of Gaussian Filtered Image "01.jpg" with k = 28



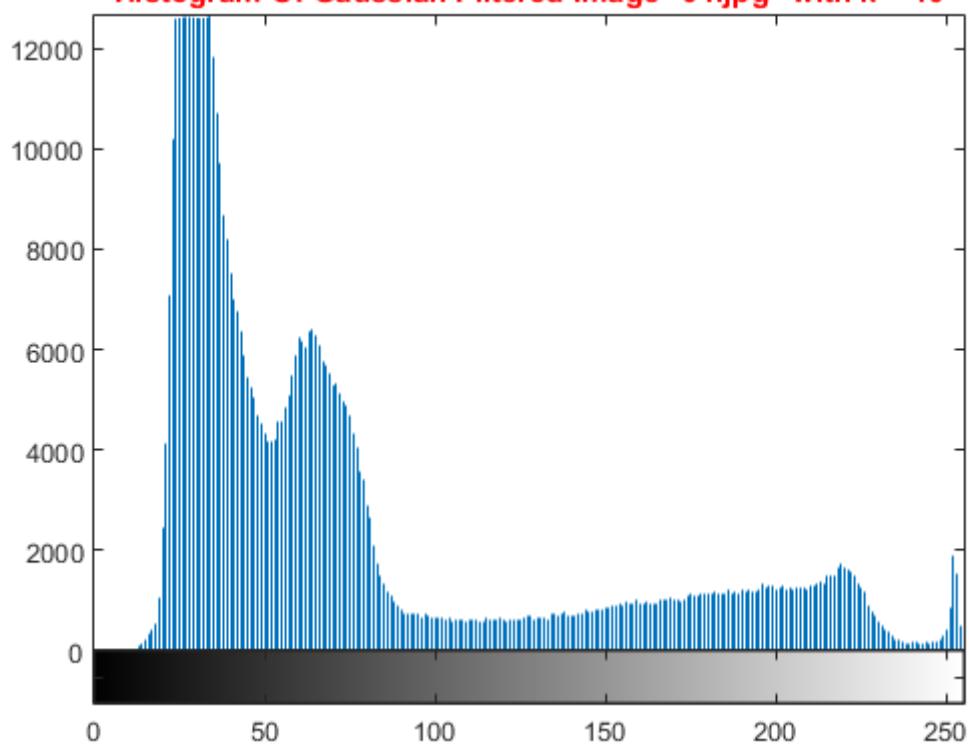
Method 2 : Absolute Value Of Fourier Transform Of Gaussian Filtered Image "01.jpg" with k = 28



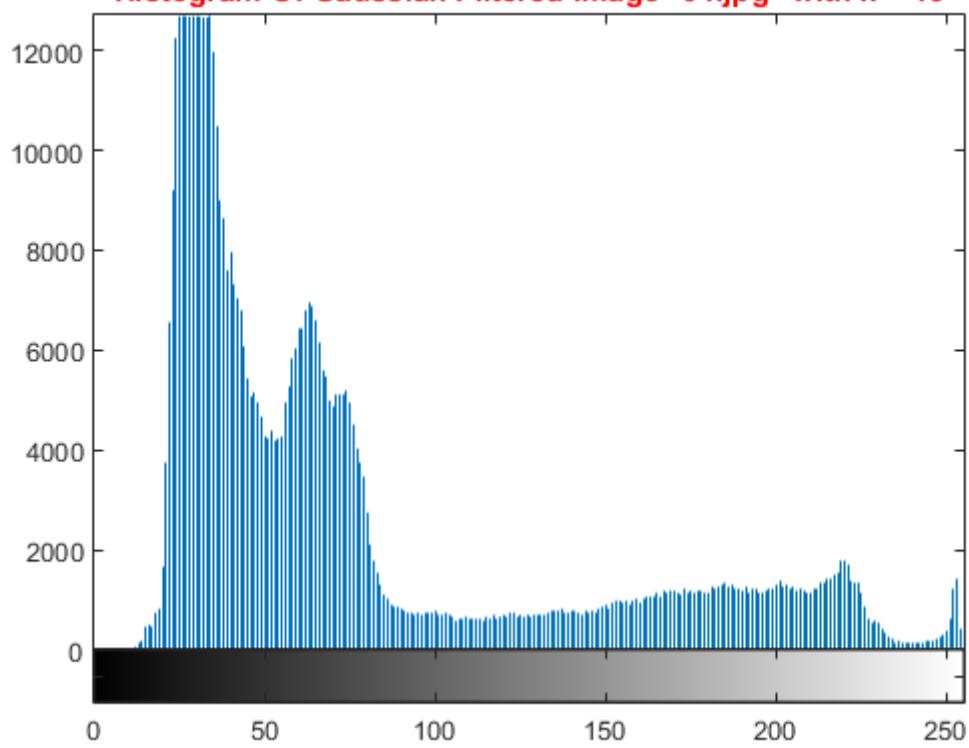
Histogram Of Gaussian Filtered Image "01.jpg" with k = 4

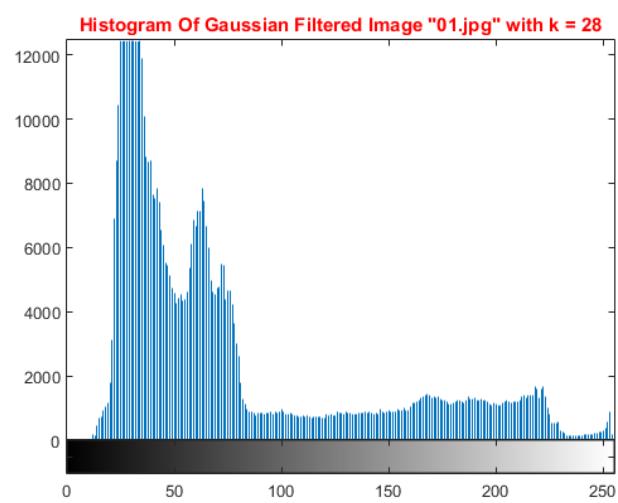
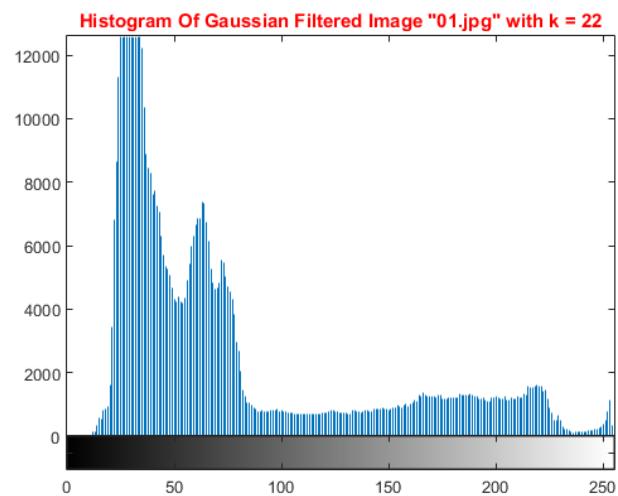


Histogram Of Gaussian Filtered Image "01.jpg" with k = 10

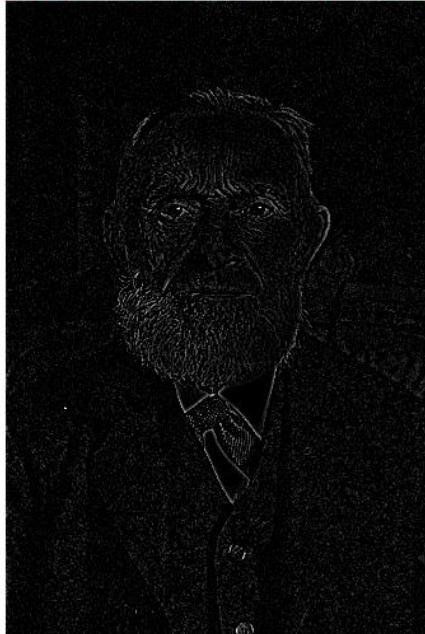


Histogram Of Gaussian Filtered Image "01.jpg" with k = 16





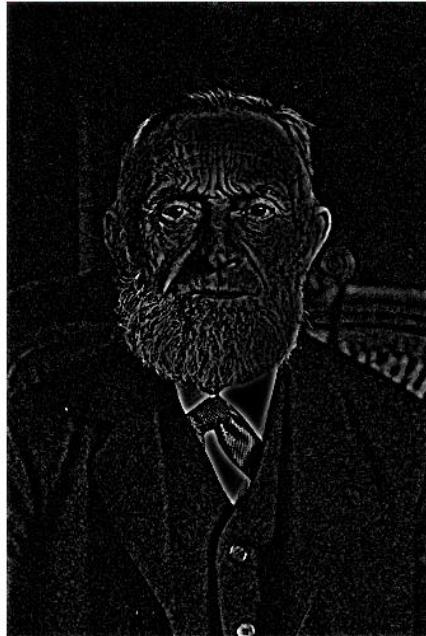
Difference Between Original & Blurred Versions of "01.jpg" for k = 10



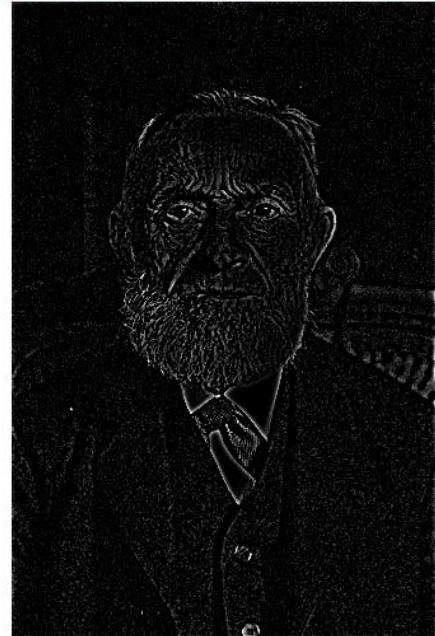
Difference Between Original & Blurred Versions of "01.jpg" for k = 4



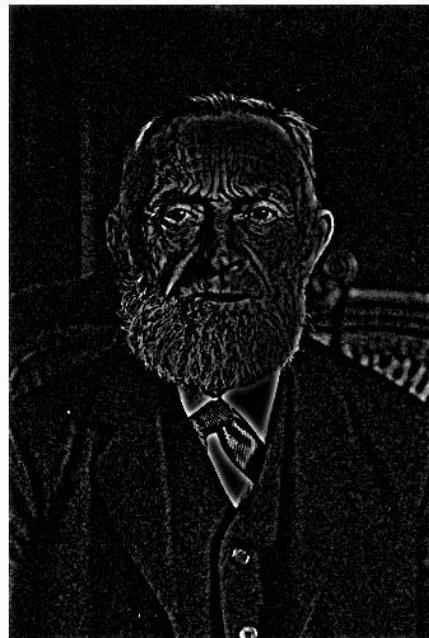
Difference Between Original & Blured Versions of "01.jpg" for k = 22



Difference Between Original & Blured Versions of "01.jpg" for k = 16

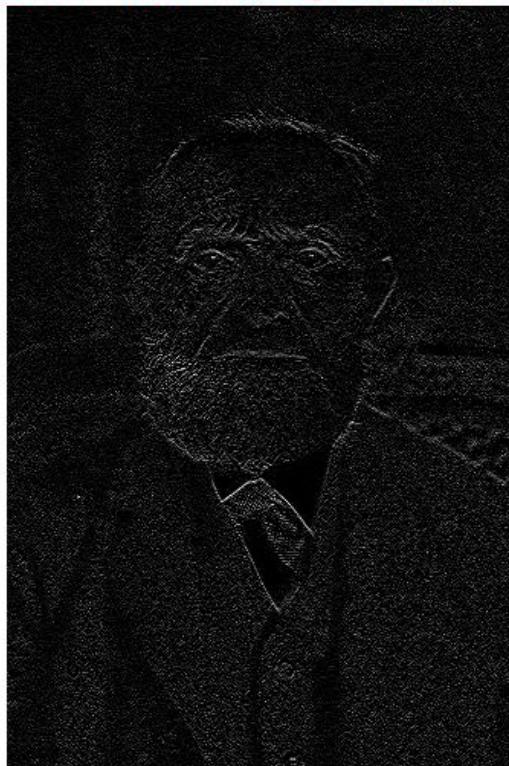


Difference Between Original & Blured Versions of "01.jpg" for k = 28

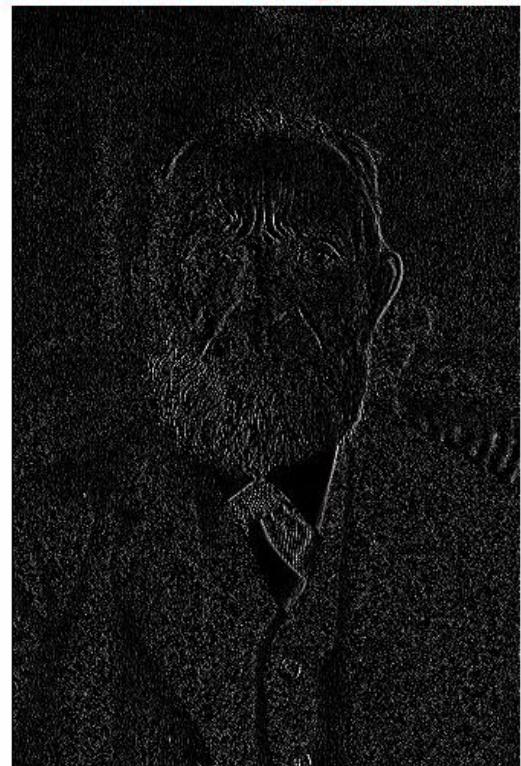


۴. توضیح کد: ابتدا باید فیلتر گاوی را طراحی کرد. این فیلتر یک پنجره مربعی و یک انحراف معیار دارد. این مقادیر با استفاده از آزمون و خطا بدست آمدند. فیلتر مطابق قبل با دستور *fspecial* و مقادیر ذکر شده طراحی می‌شود. سپس برای مشتق جزیی در جهت α باید مقدار دو پیکسل افقی متواالی و برای مشتق جزیی در جهت β باید مقدار دو پیکسل عمودی متواالی از هم کم شود. با دستور *imfilter* این کار انجام می‌گیرد. در ورودی اول آن همان ماتریس فیلتر گاوی و در ورودی دوم بردار مناسب برای اعمال مشتق جزیی داده می‌شود. حال ماتریس‌های لازم برای مشتق جزیی گرفتن را در اختیار داریم. در ادامه تصویر را با این فیلترهای مشتق جزیی فیلتر می‌کنیم و نتایج را نشان می‌دهیم.(در هنگام نمایش برای وضوح بیشتر مقادیر تصویر را در ۳ ضرب می‌کنیم). برای افزایش *contrast* نیز از دستور *imadjust* استفاده می‌کنیم. مقادیر لازم برای تنظیم *contrast* را با آزمون و خطا مشخص کردیم.

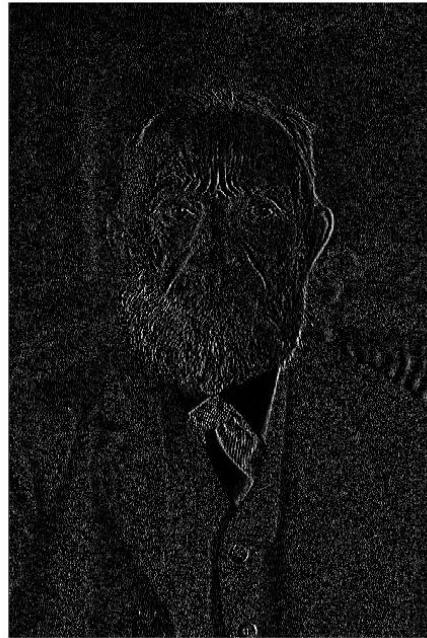
First Derivative Of Blurred Image With Respect To Y



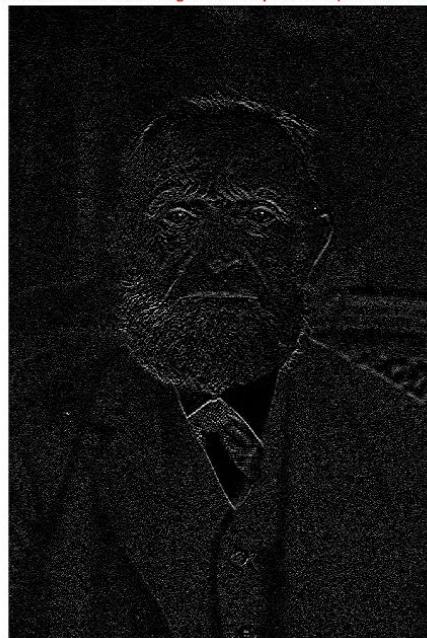
First Derivative Of Blurred Image With Respect To X



First Derivative Of Blurred Image With Respect To X (Contrast Increased)



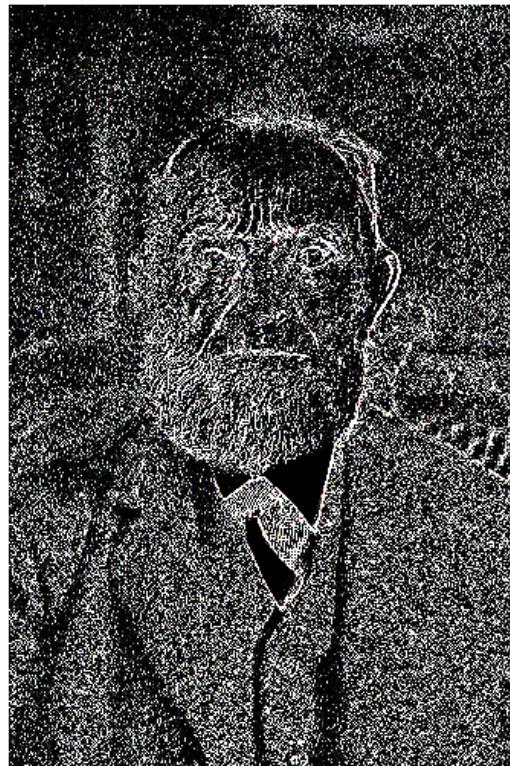
First Derivative Of Blurred Image With Respect To Y (Contrast Increased)



نتیجه نمایش با ضرب کردن و افزایش *contrast* تقریباً شبیه به هم است.

۵. توضیح کد: همان طور که می‌دانیم مجدد گرادیان برابر حاصل جمع مشتق جزئی در جهت x و y است. بنابراین برای محاسبه گرادیان، مقدار *double* شده تصاویر مشتق‌گرفته شده را به توان دو رسانده و با هم جمع می‌کنیم. جذر آن را حساب کرده و برای مناسب شدن مقادیر برای نمایش به وسیله دستور *imshow* از *uint8* استفاده می‌کنیم تا این مقادیر به اعداد صحیح بدون علامت ۸ بیتی تبدیل شوند. نهایتاً اندازه گردیان را نمایش می‌دهیم.

Magnitude Of Gradient Of image "01.jpg"



۶. در ابتدا به نحوه کار الگوریتم سوبل بصورت کلی می‌پردازیم: مبنای کار این الگوریتم، استفاده از گرادیان تصویر است. به این صورت که این الگوریتم با استفاده از دو فیلتر مشتق افقی و عمودی (که در زیر با نام‌های G_x و G_y آورده شده‌اند)، مشتق‌های جزئی و سپس اندازه گرادیان تصویر را محاسبه می‌کند؛ محاسبه‌ی اندازه گرادیان با استفاده از مشتق‌ات جزئی، شامل دو روش دقیق و تقریبی است که این دو روش در زیر آورده شده‌اند:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$\|G\| = \sqrt{Gx^2 + Gy^2}$$

روش دقیق محاسبه اندازه گرادیان:

$$|G| = |Gx| + |Gy|$$

روش تقریبی محاسبه اندازه گرادیان:

واضح است که روش تقریبی حجم محاسبات کمتری دارد.(متلب از روش تقریبی برای محاسبه اندازه گرادیان استفاده می کند)

مزیت های روش سوبل نسبت به الگوریتم مشابه خود یعنی Roberts Cross یکی تاثیرپذیری کمتر از نویز (بخاطر ماسک بزرگتر) بوده و دیگری ضخامت و وضوح بیشتر لبه هاست.

پس از محاسبه اندازه گرادیان، آستانه مناسبی روی تصویر اعمال می شود و پیکسل های باشدت روشنایی کمتر از مقدار آستانه به صفر و بقیه به 1 نگاشته می شوند، لذا تصویر خروجی یک تصویر باینری خواهد بود.

در اینجا ما با استفاده ازتابع edge در متلب، ابتدا لبه های تصویر 01.jpg را با مقدار آستانه ی پیشفرض رسم کردہ ایم (مقدار آستانه پیشفرض 0.1112 درنظر گرفته شده است). سپس برای نمونه مقدار آستانه را 50٪ افزایش داده ایم که همانطور که مشاهده می شود با افزایش مقدار آستانه، تعداد بیشتری از ناحیه های کاندیدای لبه بودن حذف می شوند؛ در دو تصویر آخر نیز صرفا لبه های افقی و عمودی را رسم کردہ ایم.

Edges Of Image 01.jpg With Sobel Edge Detector



Vertical Edges Of Image 01.jpg With Sobel Edge Detector



Horizontal Edges Of Image 01.jpg With Sobel Edge Detector



Edges Of Image 01.jpg With Sobel Edge Detector With Threshold = 1.667379e-01



قسمت چهارم: ترکیب تصاویر

۱. توضیح کد: ابتدا تصویر دوم که تا اینجا خوانده نشده بود، خوانده می‌شود. سپس تبدیل فوریه هر دو تصویر با تابعی که ساخته‌بودیم گرفته می‌شود. حال باید تبدیل فوریه تصاویر ۳ و ۴ ساخته شود. برای این منظور از رابطه زیر استفاده می‌کنیم:

$$\mathcal{F}\{img(3)\} = |\mathcal{F}\{img(1)\}| \times \angle \mathcal{F}\{img(2)\}$$

$$\mathcal{F}\{img(4)\} = |\mathcal{F}\{img(2)\}| \times \angle \mathcal{F}\{img(1)\}$$

نهایتاً با توجه به تابعی که نوشته‌ایم، باید ابتدا از این تبدیلات فوریه جدید *ifftshift* و سپس *ifft2* گرفته تا به تصاویر مورد نظر برسیم. در آخر برای اینکه قابل نمایش بهوسیله دستور *imshow* باشند، لازم است این مقادیر که از جنس *double* هستند، به مقادیر صحیح تبدیل شوند. لذا از دستور *uint8* استفاده می‌کنیم تا به اعداد *unsigned* و ۸ بیتی تبدیل شوند. نتایج در تصاویر زیر مشاهده می‌شوند. همان‌طور که از قبل می‌دانستیم، اثر غالب در تعیین یک تصویر به فاز مربوط است و وقتی اندازه‌ها از تصویری دیگر باشند به جز خراب شدن کیفیت تصویر نتیجه مؤثر دیگری نمی‌بینیم و اطلاعات اصلی شکل حفظ می‌شود.

Image 4 : Magnitude Of Image "02.jpg" & Phase Of Image "01.jpg"

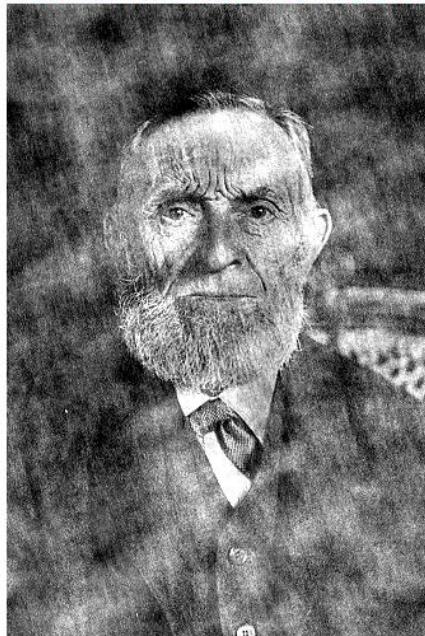


Image 3 : Magnitude Of Image "01.jpg" & Phase Of Image "02.jpg"



۲. همان‌طور که می‌دانیم در این‌گونه تصاویر محتوای فرکانس پایین یک تصویر با محتوای فرکانس بالا تصویر دیگر ترکیب می‌شود.

توضیح کد: باید با یک فیلتر پایین‌گذر فرکانس‌های بالای هر تصویر را جدا کنیم. برای این کار از فیلتر پایین‌گذر گاووسی استفاده می‌کنیم. با دستور $fSpecial$ از فیلتر گاووسی با پنجره به طول ۱۰۰ و انحراف معیار ۱۰ استفاده می‌کنیم.(هر دوی این اعداد با آزمون و خطا و امتحان کردن مقادیر مختلف برای بدست آوردن نتیجه مطلوب بدست آمدند). سپس هر تصویر را با ماتریس خروجی دستور بالا فیلتر می‌کنیم تا فرکانس‌های پایین تصویرها بدست آیند. بعد از آن این فرکانس‌های پایین را از تصویر اصلی کم می‌کنیم تا فرکانس‌های بالای تصاویر بدست آیند. برای بالاتر رفتن کیفیت تصویر *Hybrid contrast*(تضاد) تصویر را با دستور *imadjust* بالا می‌بریم. مقادیر داده شده به عنوان ورودی نیز با آزمون و خطا بدست آمدند. هنگام نمایش نیز مقادیر تصاویر در ۰.۶ ضرب شده‌اند تا تصویر *Hybrid* بهتری بدست آید. این نتایج نیز در تصاویر زیر قابل مشاهده‌اند.

Hybrid Image : "01.jpg" Lowpassed & "02.jpg" Highpassed



Hybrid Image : "02.jpg" Lowpassed & "01.jpg" Highpassed



۳. برای انجام این کار ابتدا برای هر پیکسل تصویر دوم، نسبت مقادیر سبز و آبی به مقدار قرمز (یعنی نسبت G/R و B/R) را می‌یابیم، از طرفی اگر مقیاس خاکستری تصویر اول را درنظر گرفته و آنرا با $I1$ نشان دهیم، مقدار $I1$ با رابطه‌ی زیر به مقادیر $R1$ و $B1$ مربوط می‌شود؛ برای حاصل شدن مطلوبمان، باید برای هر پیکسل تصویر اول، این مقدار ثابت بماند ولی نسبت مقادیر R و G و B آن با نسبت‌های متناظر از تصویر دوم برابر شود.

$$I1 = 0.2989 \times R1 + 0.5870 \times G1 + 0.1140 \times B1$$

لذا اگر نسبت مقادیر سبز و آبی به قرمز برای تصویر دوم را به ترتیب با $k1$ و $k2$ نشان دهیم، مقادیر R و G و B تصویر 01.jpg جدید برای هر پیکسل بصورت زیر بدست می‌آیند :

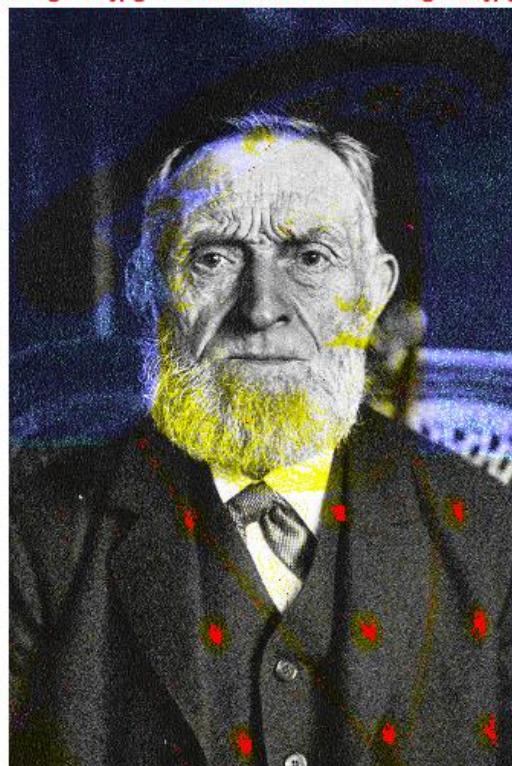
$$R1 = \frac{I1}{0.2989 + 0.5870 \times k1 + 0.1140 \times k2}$$

$$G1 = \frac{I1 \times k1}{0.2989 + 0.587 \times k1 + 0.114 \times k2}$$

$$B1 = \frac{I1 \times k2}{0.2989 + 0.587 \times k1 + 0.114 \times k2}$$

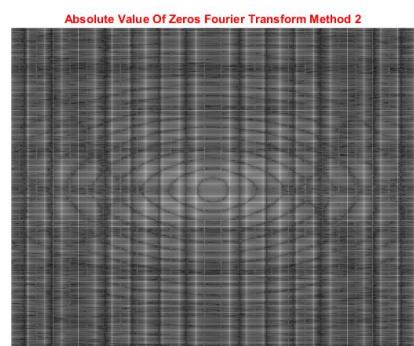
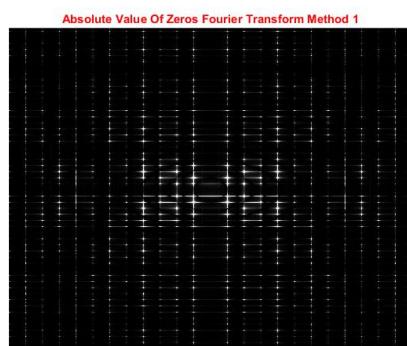
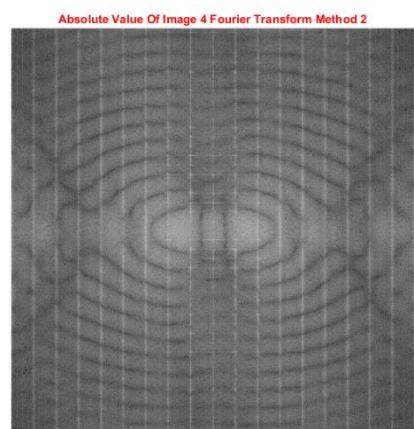
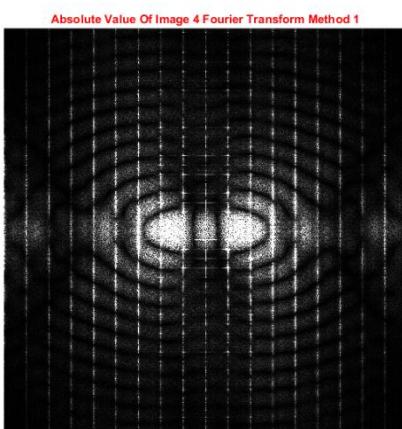
لذا تصویر نهایی بصورت شکل زیر خواهد شد :

Image 01.jpg That Is Colored With Image 02.jpg

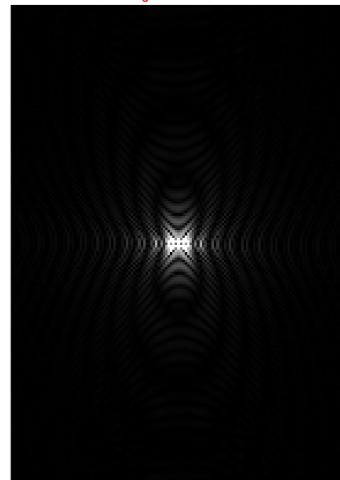


قسمت پنجم: شمارش اعداد

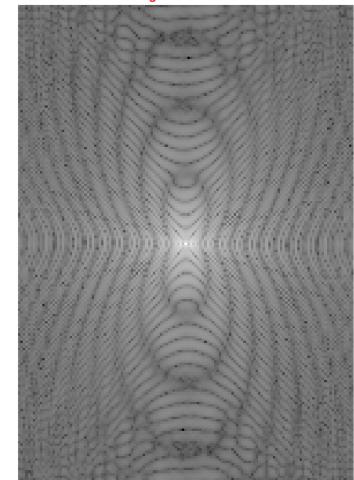
۱. با هر دو روش معمولی که تا به اینجا تبدیل فوریه را رسم می‌کنیم، برای بررسی دقیقتر و کسب اطلاعات بیشتر، تصاویری با تنها یک عدد ۱ و یک عدد صفر با همان فونت عکس داده شده و همچنین تصاویری با همان تعداد سطر و ستون تصویر داده شده که یکی همه درایه‌هایش ۱ و دیگری همه درایه‌هایش صفر باشد ساختیم. (این تصاویر نیز پیوست شده‌اند). اندازه تبدیل فوریه این تصاویر پس از رسم اندازه تبدیل فوریه تصویر اصلی نشان داده می‌شود.



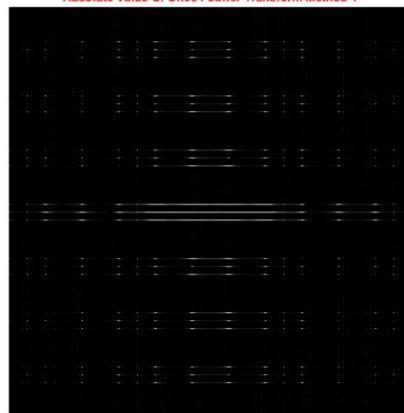
Absolute Value Of a Single Zero Fourier Transform Method 1



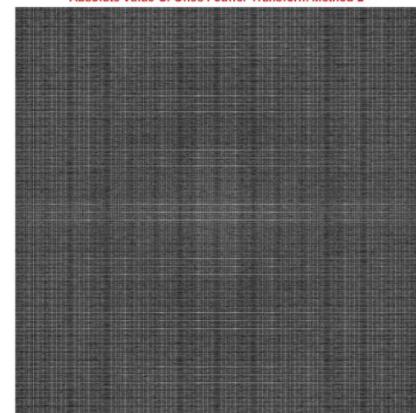
Absolute Value Of a Single Zero Fourier Transform Method 2



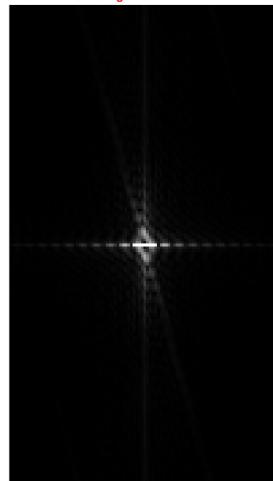
Absolute Value Of Ones Fourier Transform Method 1



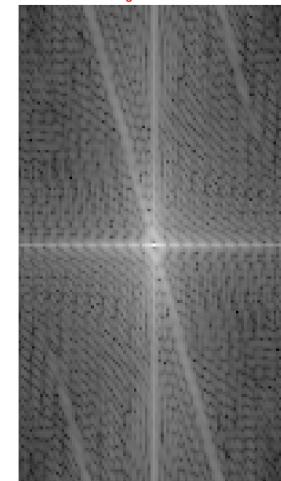
Absolute Value Of Ones Fourier Transform Method 2



Absolute Value Of a Single 1 Fourier Transform Method 1



Absolute Value Of a Single 1 Fourier Transform Method 1

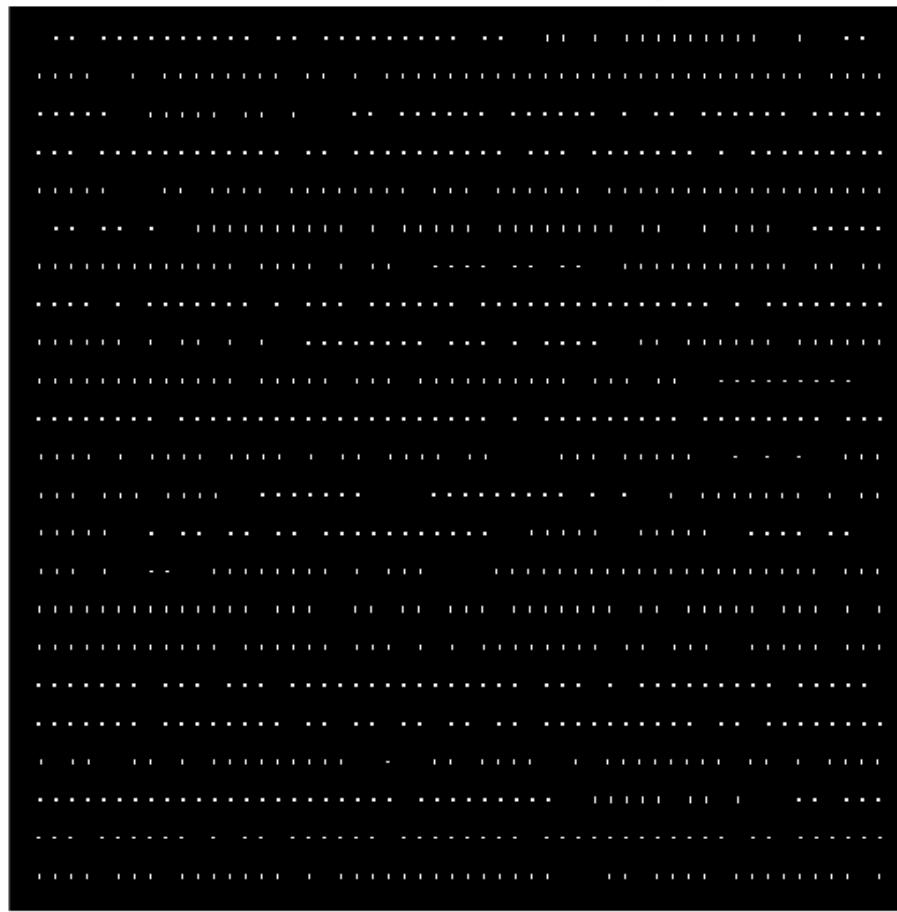


ابتداً ترین مواردی که قابل تشخیص است، تعداد سطرهای تصویر و تعداد کاراکترهای هر سطر است. همانطور که می‌دانیم، وقتی از بالا به پایین می‌آییم به نظر می‌رسد بین سطرهای پر و خالی یک نوع نوسان وجود دارد و چون در جهت عمودی حرکت می‌کنیم این هارمونیک خود را در جهت افقی در تصویر تبدیل فوریه نمایش می‌دهد. در واقع اگر در تصویر تبدیل فوریه از چپ به راست حرکت و کنیم تعداد خطوط عمودی روشن را بشماریم به تعداد سطرهای دارای متن می‌رسیم که می‌توان با شمارش به عدد 23 (تعداد سطرها) رسید. در هر سطر نیز با حرکت افقی بین حروف هارمونیک‌هایی به چشم می‌خورد که برای مثال برای تصویر تمام 1، با شمارش خطهای افقی در تصویر تبدیل فوریه، به عدد 54 (تعداد کاراکترهای هر سطر) می‌رسیم. بنابراین خطوط عمودی و افقی پشت زمینه مربوط به سطرها و ستون‌ها هستند.

نکته دیگری راجع به تصویر که به نظر می‌رسد غالب بودن اثر صفرها است. همانطور که در قسمت بعد شمارش می‌شود تعداد صفرها حدوداً 4 برابر یک‌هاست و تبدیل فوریه غالباً مشابه تصویر تبدیل فوریه عکسی است که همه‌اش صفر است. در مورد تصویر تبدیل فوریه نیز می‌توان این‌گونه توجیح کرد که صفر مانند یک بیضی است که از تکه‌خطهای کوچک به همپیوسته تشکیل شده‌است. هر خط کوچک در تبدیل فوریه یک سینک ایجاد می‌کند و می‌توانیم مشاهده کنیم که برای مجموعه این قطعه‌خطهای تشکیل دهنده صفر، سینک در همه جهات امتداد دارد.

۲. توضیح روش: شمردن صفرها همان‌طور که از قبل می‌دانیم برای یافتن سیگنال‌های شبیه به هم می‌توان از کراس کورریلیشن استفاده کرد. مشابه یافتن اکو، در اینجا می‌توانیم برای یافتن صفرها از خود تصویر یک صفر جدا کرده و کراس کورریلیشن آن را با کل تصویر محاسبه کنیم. در اطراف صفرهای تصویر ماکزیمم موضعی خواهیم داشت. می‌توانیم تصویر نتیجه کراس کورریلیشن را مشاهده کنیم. نتیجه مانند تصویر زیر نقطه‌هایی روشن است. البته حدی انتخاب کرده‌ایم که اگر نتیجه $xcorr2$ از عددی بیشتر باشد آن نقطه روشن شود. (این حد را پس از امتحان کردن مقادیر متعدد انتخاب کردیم. به گونه‌ای که نه نقاط روشن اضافی داشته باشیم و نه نقاط روشنی از قلم بیفتند. همه نقاط روشن به محل صفرها اشاره می‌کنند). نتیجه تصویری مانند شکل زیر است.

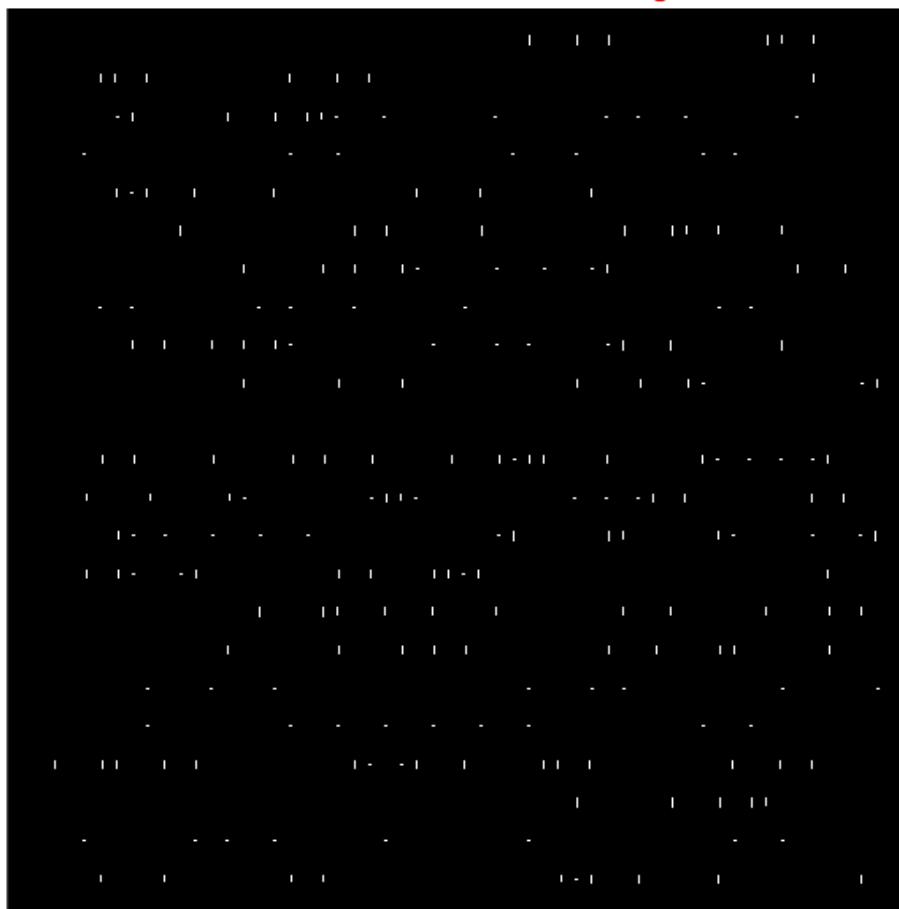
Result Of Cross Correlation With A Single Zero



در ادامه بایست به روشی این نقاط را بشماریم. ما بدین صورت عمل می‌کنیم؛ با دستور \max یک سطر طولانی بدست می‌آوریم که در هر ستون آن مشخص شده در چندمین سطر پیکسلی تصویر مقدار ماکزیمم آن ستون پیکسلی قرار دارد. برای مثال این سطر به تعداد پیکسل‌های افقی تصویر، ستون خواهد داشت. اگر در ستون ۴۰۰ این سطر، عدد ۱۴۰ نوشته شده باشد به این معنی است که در ماتریس تصویر(مبدا بالا چپ است)، ۱۴۰ پیکسل به سمت پایین از مبدا و ۴۰۰ پیکسل به سمت راست برویم تا به پیکسل دارای ماکزیمم مقدار در بین پیکسل‌های ستون ۴۰۰ تصویر برسیم. حال در ادامه، یک حلقه *for* خواهیم داشت. این حلقه هر بار بر روی یک ستون تصویر عمل می‌کند. در این حلقه برای هر ستون، پیکسل دارای ماکزیمم مقدار یافت شده، اگر این ماکزیمم از حدی بیشتر باشد، اطراف آن صفر می‌شود(چون هر نقطه روشن مربوط به وجود صفر، از تعدادی پیکسل تشکیل شده و باید همه آن‌ها صفر شوند تا در ادامه شمرده نشوند) و به شمارنده یکی اضافه می‌شود. سپس ماتریس ماکزیمم‌ها دوباره گرفته می‌شود و در حلقه *while* چک می‌شود که آیا در این ستون از پیکسل‌ها، مقدار روشن دیگری داریم یا نه. اگر داشته باشیم همین روند ادامه می‌یابد تا در این ستون هیچ پیکسل با مقدار زیاد(روشن) باقی نماند. حال به سراغ ستون بعدی می‌رویم. توجه می‌کنیم که آن

دسته از نقاط مربوط به صفری که با توجه به محدودهای که صفر کردیم حذف شدند، دیگر در این ستون از پیکسل‌ها جای ندارند و اشتباهات دوباره محاسبه نمی‌شوند. همان روند برای این ستون تکرار شده و اطراف صفرهای یافت شده خاموش می‌شود و به سراغ ستون بعدی می‌رویم. این روند تا آخرین ستون ادامه می‌یابد. شمردن یک‌ها نیز به همین ترتیب است و فقط باید یک تصویر ۱ از تصویر داده شده جدا کنیم و با تصویر اصلی کراس کورریلیشن بگیریم. نتیجه آن هم در تصویر زیر مشخص است.(البته متوجه شدیم که اگر حد روشن کردن نقطه‌ها را بالا در نظر بگیریم، اگرچه در تصویر مشاهده شده توسط ما یا همان تصویر زیر دیده نشوند ولی با بزرگنمایی می‌توان آن‌ها را دید و این بالاگرفتن حد منجر به شمارش دقیق‌تر و کم‌کردن احتمال شمارش نقطه‌های اشتباه می‌شود. بنابراین نقطه‌های روشن تصویر زیر تمام ۱ ها را مشخص نمی‌کند. بلکه باید در تصویری که در مطلب باز می‌شود زوم کنید و روشن شدن محل بقیه ۱ ها را ببینید. این مشاهده نشدن اهمیتی ندارد چون در شمارش همه این نقاط لحاظ نمی‌شوند).

Result Of Cross Correlation With A Single One



توضیح کد:

صفرها: ابتدا با بازی کردن با اعداد سطر و ستون تصویر یک صفر تنها مانند تصویر زیر انتخاب می‌شود.

0

در ادامه با توجه به اینکه ترجیح می‌دهیم نقطه‌های حاوی اطلاعات ما روشن باشند(صفر و یک‌ها) تصویر صفر را متمم می‌کنیم. همین‌طور تصویر کل را نیز متمم می‌کنیم. سپس مقادیر تصویر صفر کوچک را به فرمت اعشاری برد و با مقیاس 255 و طول تصویر برای اعمال کردن بر کل تصویر آماده می‌کنیم. سپس نتیجه $xcorr2$ بدست می‌آید و بعد نرمالایز می‌شود. پس از آن اگر برای هر پیکسل نتیجه کراس کورریلیشن از حدی بیشتر باشد در 255 ضرب می‌شود تا کاملاً روشن گردد. سپس نتیجه این عملیات در تصویری نشان داده می‌شود. بعد از آن مقدار روشنایی ماکزیمم هر ستون در A و شماره سطر آن ماکزیمم مربوط به هر ستون در B ذخیره می‌شوند و شمارنده با صفر مقداردهی اولیه می‌شود. نهایتاً وارد حلقه for ای می‌شویم که در بالا توضیحات آن داده شد. حد روشنایی نقاط ماکزیمم برای جلوگیری از بروز خطا زیاد انتخاب شد(200). محدوده خاموش کردن اطراف نیز با بازی کردن با مقادیر و یافتن فاصله نوعی دو نقطه روشن مربوط به دو صفر متوالی انتخاب شدند تا خاموش کردن یک صفر بر صفر دیگری اثر نگذارد. نهایتاً تعداد صفرها در *command window* نمایش داده می‌شود.

یک‌ها: روند کاملاً مشابهی برای یک‌ها اجرا می‌شود. نتیجه نهایی به شرح زیر است:

Number Of Zeros : 992

Number Of Ones : 250