

باسم‌هه تعالی



## دانشگاه صنعتی شریف

دانشکده مهندسی برق

## درس سیگنال‌ها و سیستم‌ها

گزارش تمرین مطلب

سری پنجم

امیرحسین افشارراد

۹۵۱۰۱۰۷۷

بهزاد منیری

۹۵۱۰۹۵۶۴

۱۳۹۷ بهار

## سؤال اول) حذف نویز در تصاویر رنگی

### چگونگی تجزیه تصاویر به کانال‌های R، G، و B و اعمال فیلتر بر روی آنها

تصاویر رنگی در متلب به صورت ماتریس‌های سه‌بعدی ذخیره می‌شوند که بعد اول و دوم ماتریس مشخصات طول و عرض تصویر را در بر دارد و بعد سوم این ماتریس، که همواره دارای ۳ درایه است، مقدار مربوط به هر کدام از کانال‌های R، G، و B را در بر دارد. می‌توان با اعمال یک میانگین وزن‌دار روی این ۳ درایه، تصویر را از حالت رنگی تبدیل به خاکستری کرد (کاری که تابع `rgb2grey` انجام می‌دهد) که هدف این تمرین چنین نیست و می‌خواهیم روی تصاویر رنگی پردازش انجام دهیم، لذا این نمایش تصاویر را بر هم نمی‌زنیم.

برای ایجاد نویز در تصاویر، از تابع `imnoise` استفاده می‌کنیم. (به جز یک مورد که در این تابع موجود نیست و به صورت دستی اعملاً می‌شود) همان طور که نویز در هر یک از لایه‌های تصویر (R، G، و B) به صورت جداگانه اعمال می‌شود، بهتر است عملیات فیلترینگ نیز جداگانه بر روی هر کدام از این ۳ کanal صورت بگیرد. به خصوص در فیلترهایی مانند `median filter` را به صورت جداگانه روی کانال‌های مختلف انجام ندهیم، نتیجه کیفیت مطلوبی را نخواهد داشت، چرا که دیتای هر کدام از کانال‌ها و مقدار میانه به شکل مستقل از یکدیگر معنا دارد. در مورد فیلترهایی مانند `wiener filter` نیز که اقدام به تخمین زدن توان نویز می‌کنیم، ترکیب نکردن این کانال‌ها می‌تواند باعث بهبود عملکرد تخمین‌گر شود، چرا که ترکیب سه ماهیت مختلف و ارائه‌یک تخمین واحد از این ترکیب سه‌گانه احتمالاً نتیجه‌ای کم کیفیت ارائه خواهد کرد.

### أنواع نویزهای مورد استفاده

#### **۱ – Additive White Gaussian Noise (AWGN)**

در این نویز، به مقدار هر کدام از پیکسل‌های تصویر، یک مقدار تصادفی اضافه می‌شود که توزیع این مقادیر تصادفی، از یک فرایند نویز سفید گوسی پیروی می‌کند.

#### **۲ – Salt & Pepper (Impulse) Noise**

در این نوع نویز، پیکسل‌های تصویر با توجه به یک توزیع احتمال دچار نویز می‌شوند، به این صورت که هر پیکسل نویزی، یکی از دو مقادیر صفر یا ۱ (مقادیر مرزی، نشان‌گر بیشترین یا کمترین مقدار ممکن برای یک پیکسل) را به خود می‌گیرد.

یک نکته در مورد اعمال این نویز (که در ابتدا برای نگارنده نیز مورد سؤال بود) آن است که شاید انتظار داشته باشیم با اعمال این نویز روی تصویر، مجموعه‌ای از نقاط سیاه و سفید (که نام نویز – فلفل‌نمکی – نویز از وجود این دو نوع نقطه‌گرفته شده است) روی تصویر ظاهر شود، در حالی که در ادامه می‌بینیم که چنین نمی‌شود و مجموعه‌ای از نقاط رنگی در شکل ظاهر می‌شوند. علت این امر، آن است که این نویز روی هر کدام از سه لایه‌ی رنگی تصویر اثر می‌کند، و این که یک، دو، یا سه رنگ دچار این نویز شوند (که دچار نویز شدن نیز خود دو حالت دارد)، موجب ایجاد نقاط رنگی متنوعی در تصویر می‌شود.

#### **۳ – Shot Noise**

این نوع نویز، که به نویز پواسون نیز مشهور است، از نظر طبیعی دارای منشأ کوانتومی و مربوط به رفتار ذرات است. از نظر ریاضیاتی و آماری، مقدار پیکسل‌ها در اثر این نویز، از یک فرایند پواسون به دست می‌آیند که پارامتر این توزیع پواسون، مرتبط با مقدار واقعی پیکسل است. در متلب (و به وسیله‌ی تابع `imnoise`، ابتداء مقادیر پیکسل‌ها با ضریب  $1e12$  بزرگ می‌شوند، سپس مقدار نویزی از یک توزیع پواسون با پارامتر مذکور به دست می‌آیند، نهایتاً با ضریب مشابه کوچک می‌شوند تا محدوده تغییرات عددی تقریباً ثابت بماند).

**٤ - Speckle (Multiplicative) Noise**

این نوع نویز، بر خلاف نویزهای additive (که مقداری تصادفی با مقادیر واقعی جمع می‌شود)، از نوع نویزهای multiplicative است، یعنی مقداری تصادفی در مقدار واقعی ضرب می‌شود. در متلب، پیکسل نویزی از رابطه‌ی  $J = I + n^*I$  به دست می‌آید، که در آن  $I$  مقدار بدون نویز است و  $n$  مقدار نویز ضرب‌شونده است که از توزیعی یکنواخت با میانگین صفر و واریانس  $\sigma^2$  (که می‌توان در ورودی تابع `imnoise` تنظیم کرد) به دست می‌آید.

**٥ - Uniform Additive Noise**

این نویز نیز مشابه با مورد اول از نوع نویزهای جمع‌شوند (additive) است که برخلاف مورد اول که توزیعی گوسی داشت، این بار نویز اضافه‌شونده از یک توزیع یکنواخت می‌آید. این نوع نویز، تنها موردنی است که در کد ضمیمه شده، بدون استفاده از تابع `imnoise` و به صورت دستی تولید شده است. برای این کار، از مولدی با توزیع  $[0.5, 0.5] \sim U$  استفاده کردیم.

**أنواع فیلترهای مورد استفاده****١ - Linear Smoothing Filter**

فیلترهای هموارساز (smoothing) خانواده بزرگی از فیلترها هستند که به صورت کلی، تغییرات شدید در سیگنال ورودی را تعدیل می‌کنند. این کار، به تعییری معادل حذف فرکانس‌های بالا و اعمال یک فیلتر پایین‌گذر است. فیلترهای هموارساز خطی نیز زیرمجموعه‌ای از این خانواده فیلترها هستند که عملیات هموارسازی را به وسیله‌ی یک تبدیل خطی انجام می‌دهند. برای این کار، می‌توان هسته‌های مختلفی برای انجام کانولوشن و اعمال فیلتر هموارساز خطی در نظر گرفت. در این پژوهه، ما از دو فیلتر هموارساز خطی استفاده کردیم که عبارتند از:

الف) هموارسازی خطی با استفاده از میانگین متحرک (هسته کانولوشن توزیع یکنواخت دارد و در این پژوهه، پنجره‌ای با ابعاد  $4 \times 4$  در نظر گرفته شده است)

ب) هموارسازی خطی گوسی (هسته کانولوشن توزیع گوسی دارد و در این پژوهه، از میانگین صفر و انحراف معیار ۲ استفاده شده است)

**٢ - Wiener Filter**

این فیلتر، با فرض ایستا بودن فرایند سیگنال و نویز، و جمع‌شونده (additive) بودن نویز، اقدام به ارائه‌ی یک تخمین از سیگنال اصلی می‌کند. این فیلتر، از تخمین‌گر MMSE (Minimum Mean Square Error) استفاده می‌کند و بر این مبنای، بهترین تخمین را از سیگنال بدون نویز ارائه می‌دهد.

از مزایای این فیلتر می‌توان به روش‌های متعدد پیاده‌سازی، مینیمم کردن خطای مجدور مربعات، و قابلیت محاسبه به صورت real-time اشاره کرد. در مورد معاوی آن نیز، می‌توان به محدودیت‌های آن از جمله خطی فرض کردن فرایند تصادفی، ارائه‌ی تنها یک تخمین نقطه‌ای، و ایستا و جمعی فرض کردن نویز اشاره کرد.

**٣ - Median Filter**

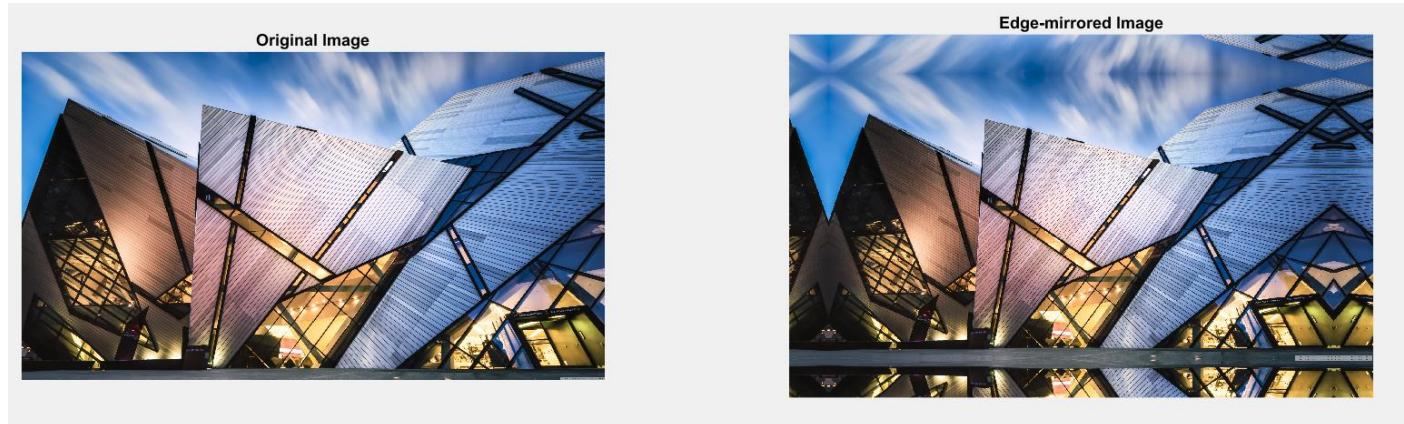
این فیلتر، با استفاده از یک پنجه‌های دلخواه، هر نقطه را با میانه‌ی نقاط همسایه‌ی خود جایگزین می‌کند و به این وسیله، اقدام رفع نویز می‌نماید.

**شرح توابع ضمیمه شده**

برای پیاده‌سازی فیلترهای مذکور، اقدام به نوشتتن چند تابع کردیم که در ادامه، مختصراً بررسی می‌شود:

**۱ – mirror\_edge**

این تابع وظیفه‌ی آن را دارد که اطراف تصویر را با پهنه‌ای دلخواه (که در ورودی تابع بر حسب پیکسل دریافت می‌شود) به صورت آینه‌ای در چهار طرف تصویر اضافه کند تا عملیات فیلترینگ در گوشه‌های تصویر، کیفیت مطلوب‌تری داشته باشد. در ادامه نمونه‌ای از عملکرد این تابع را در کنار تصویر اصلی مشاهده می‌کنید.

**۲ – remove\_edge**

این تابع وظیفه‌ای بر عکس تابع قبلی دارد و بعد از انجام فیلترینگ مورد استفاده قرار می‌گیرد، تا قسمت اصلی تصویر در خروجی ارائه شود و حواشی آن بریده شود.

**۳ – mymedfilt**

این تابع، پیاده‌سازی median filter است که با استفاده از پنجره‌ای با ابعاد  $n \times n$  (که  $n$  از ورودی و بر حسب پیکسل خوانده می‌شود)، این فیلتر را پیاده‌سازی می‌کند.

**۴ – limit\_holder**

این تابع، یک تابع کمکی برای پیاده‌سازی median filter است که در پیاده‌سازی کد مطلب مربوط به لبه‌های عکس کمک می‌کند. (این تابع صرفاً یک ابزار کمکی در پیاده‌سازی کد است، برای بررسی بیشتر می‌توانید کد آن را مشاهده کنید)

**نحوه گزارش خروجی**

در ادامه، خروجی‌های کد مطلب را گزارش می‌کنیم. عملیات اعمال نویز و فیلترینگ بر روی ۳ تصویر انجام شده است. ترتیب خروجی‌ها به این شکل است که ابتدا تصویر اصلی نمایش داده می‌شود. سپس تصویر ناشی از هر نوع نویز، و خروجی هر چهار نوع فیلتر اعمال شده بر روی تصویر نویزی نمایش داده می‌شود، سپس به سراغ نویز بعدی می‌رویم. (یعنی ترتیب تصاویر بر حسب نویزهای است، هر نویز و تمامی روش‌های رفع آن بررسی می‌شود، سپس به بررسی نویز بعدی پرداخته می‌شود)  
گزارش فوق برای ۳ تصویر ارائه می‌شود، سپس به نتیجه‌گیری و جمع‌بندی می‌پردازیم.

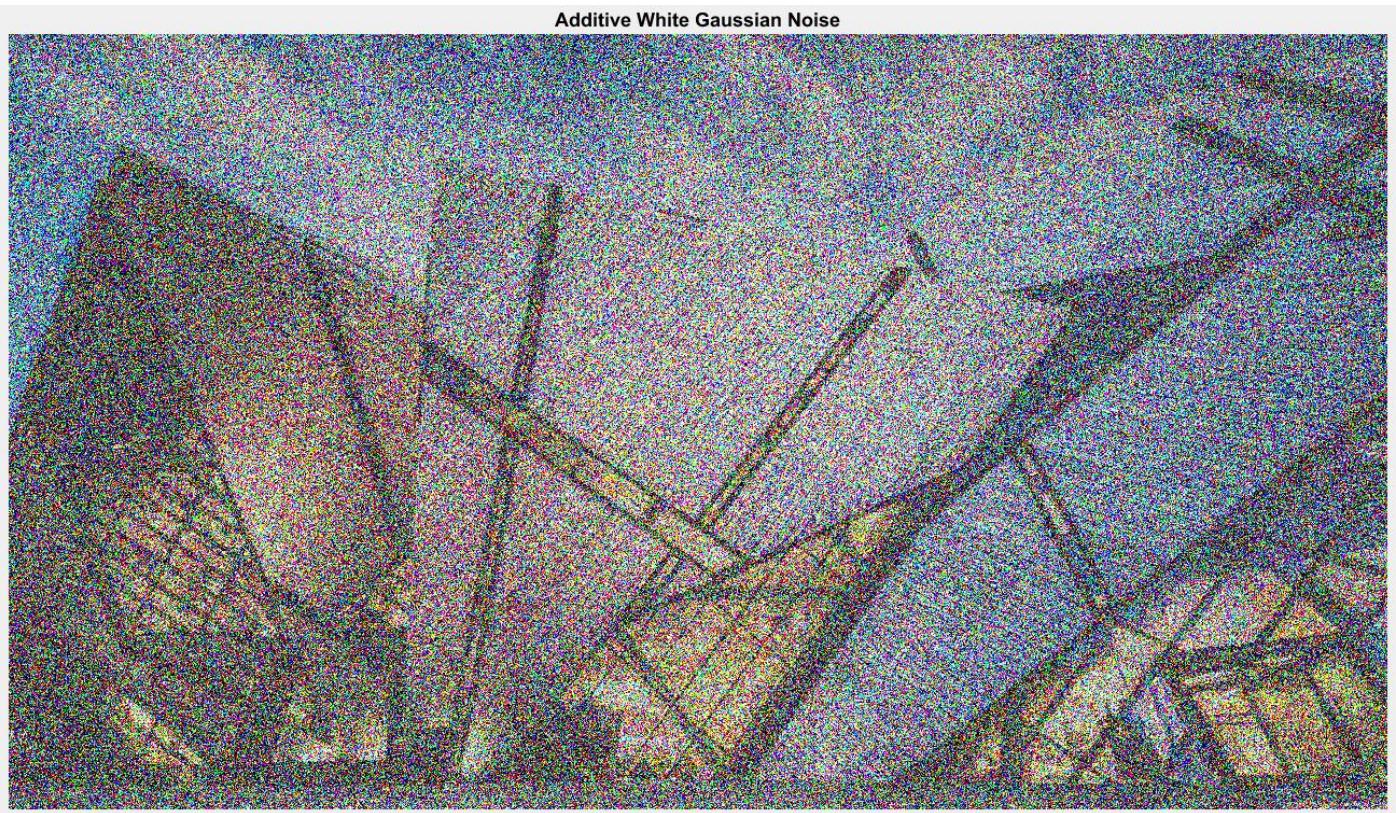
تصویر اول، بدون نویز

Original Image



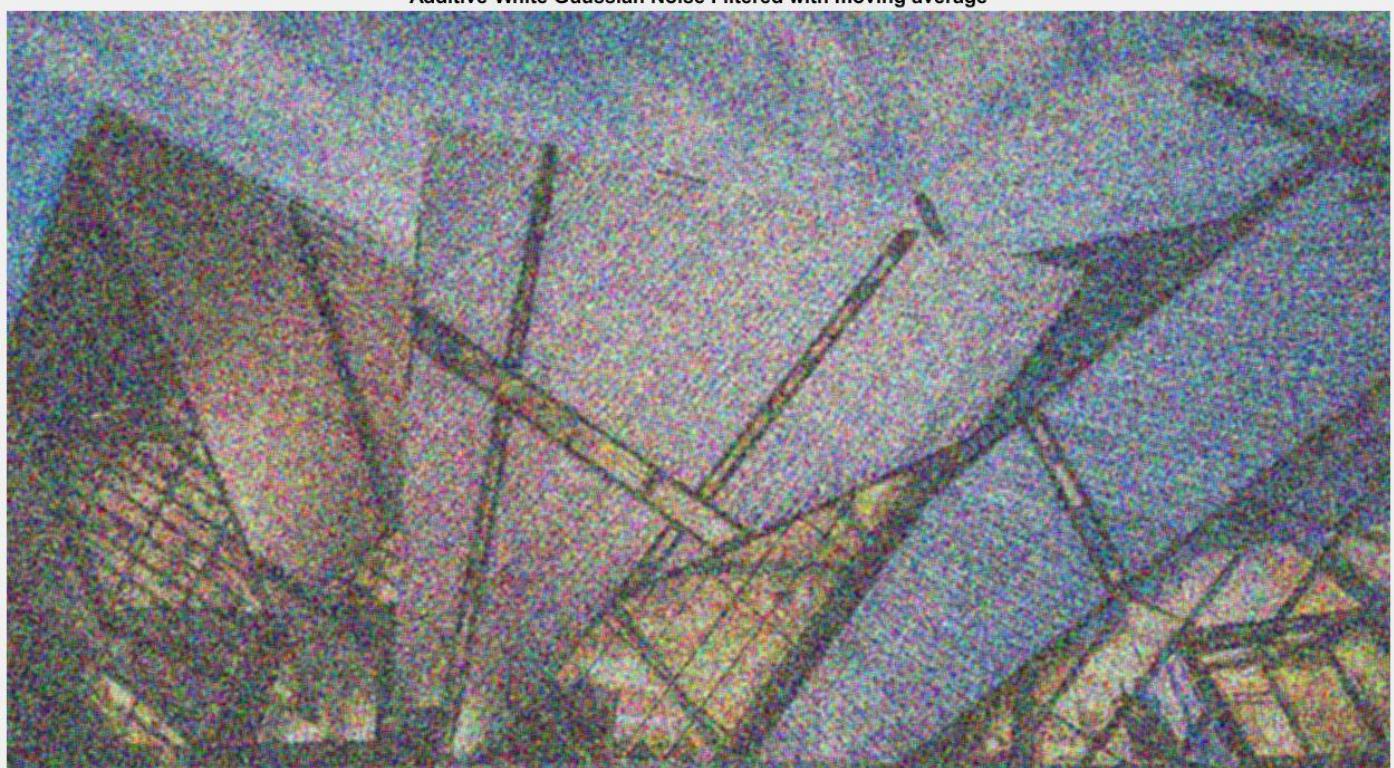
تصویر اول، نویز جمع‌شونده گوسی

Additive White Gaussian Noise



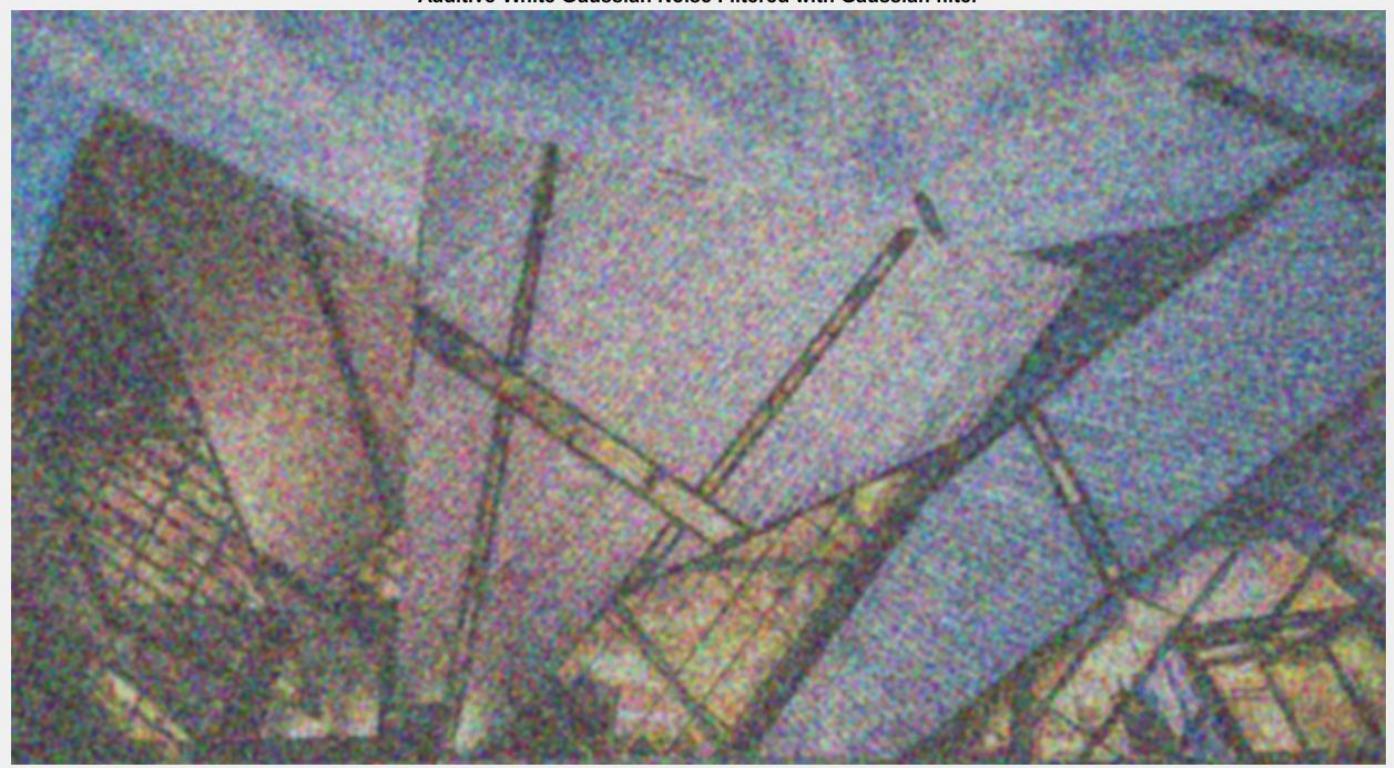
تصویر اول، نویز جمع‌شونده گوسی، فیلتر هموارساز میانگین متحرک

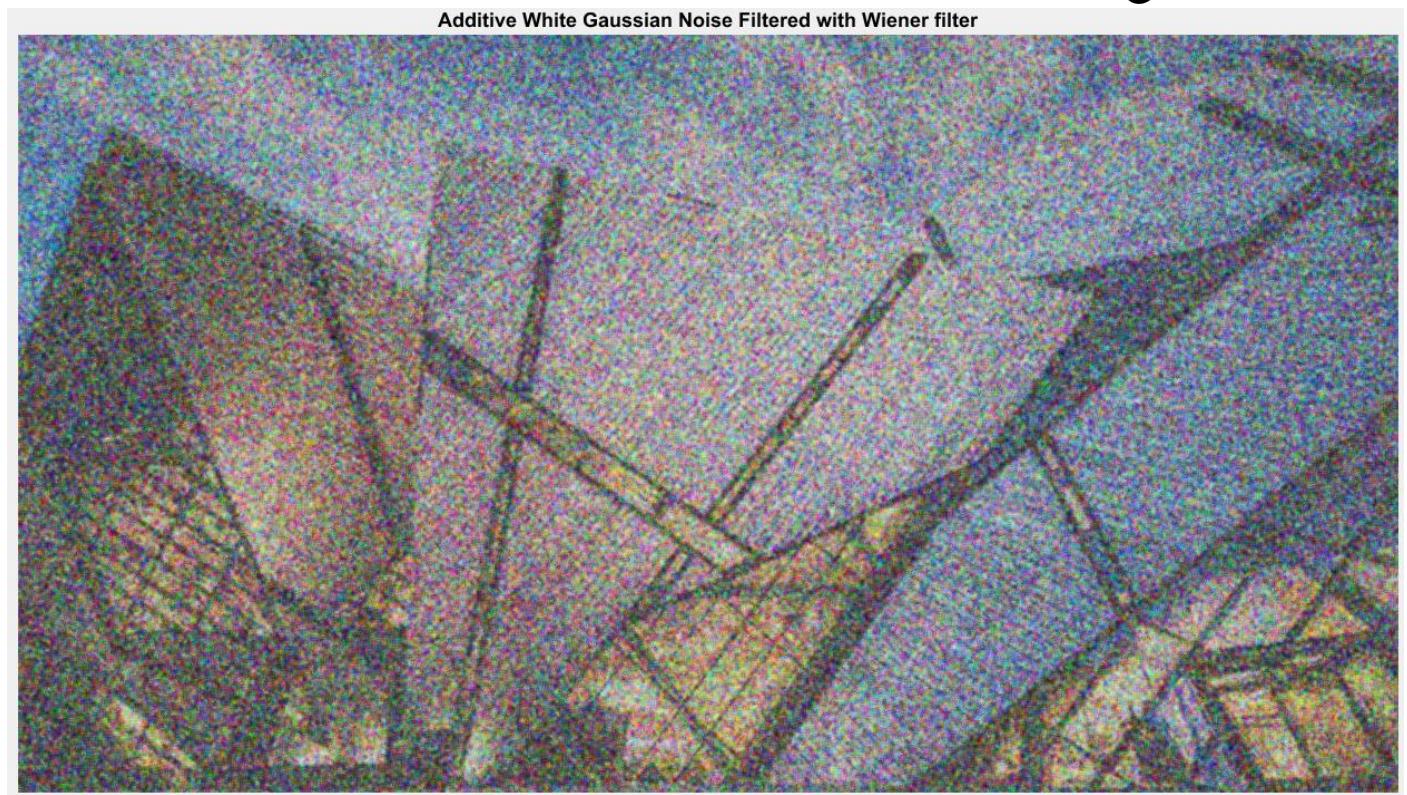
Additive White Gaussian Noise Filtered with moving average



تصویر اول، نویز جمع‌شونده گوسی، فیلتر هموارساز گوسی

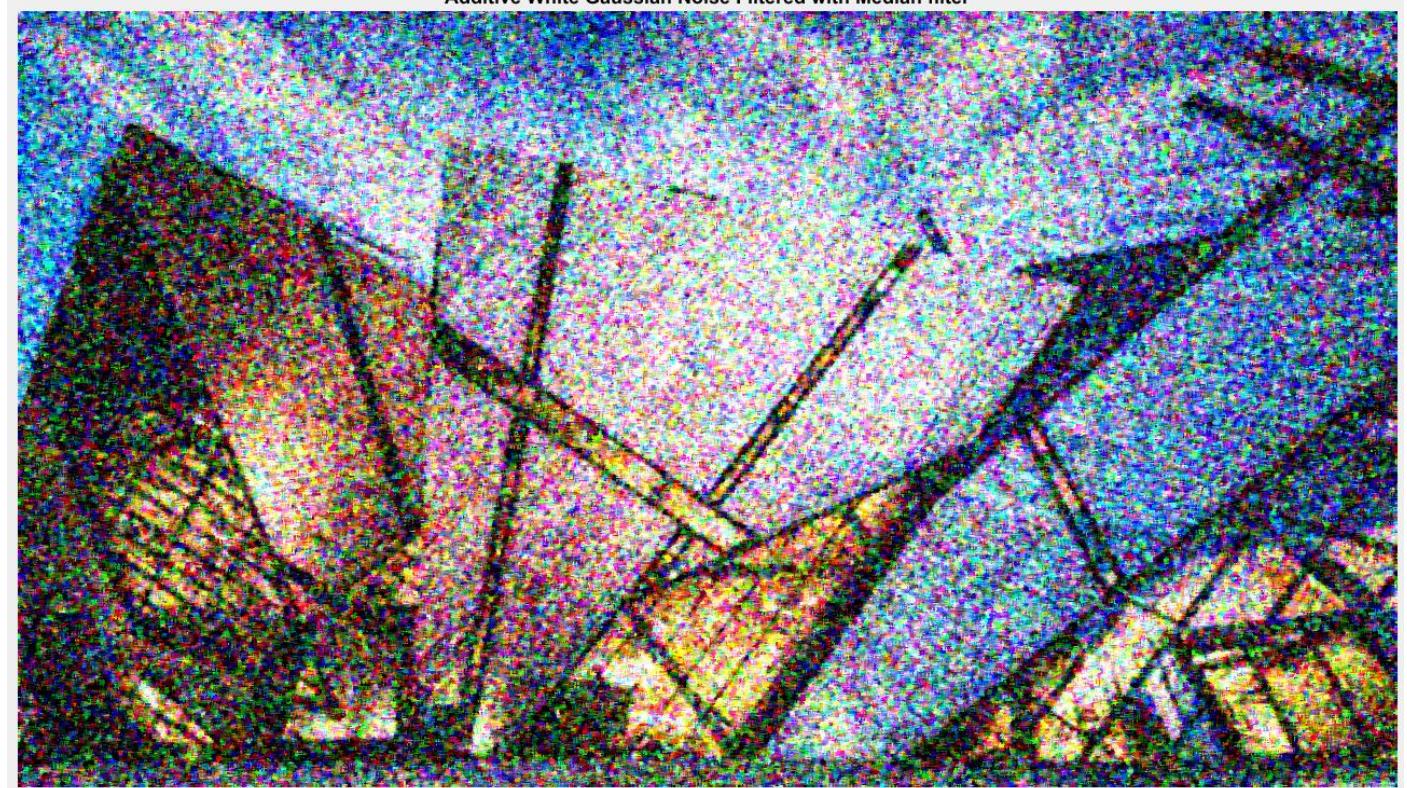
Additive White Gaussian Noise Filtered with Gaussian filter



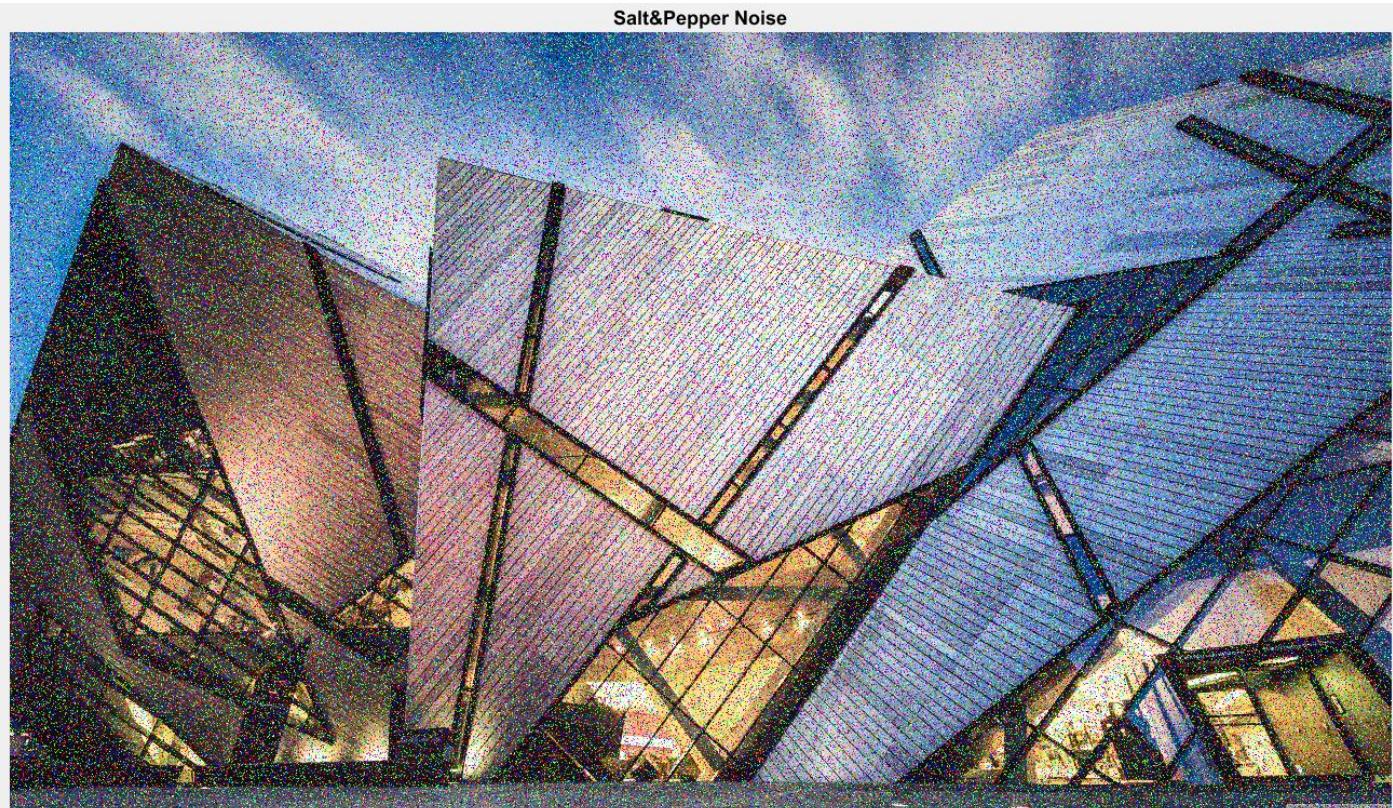


تصویر اول، نویز جمع‌شونده گوسی، فیلتر میانه

Additive White Gaussian Noise Filtered with Median filter



Salt&Pepper Noise



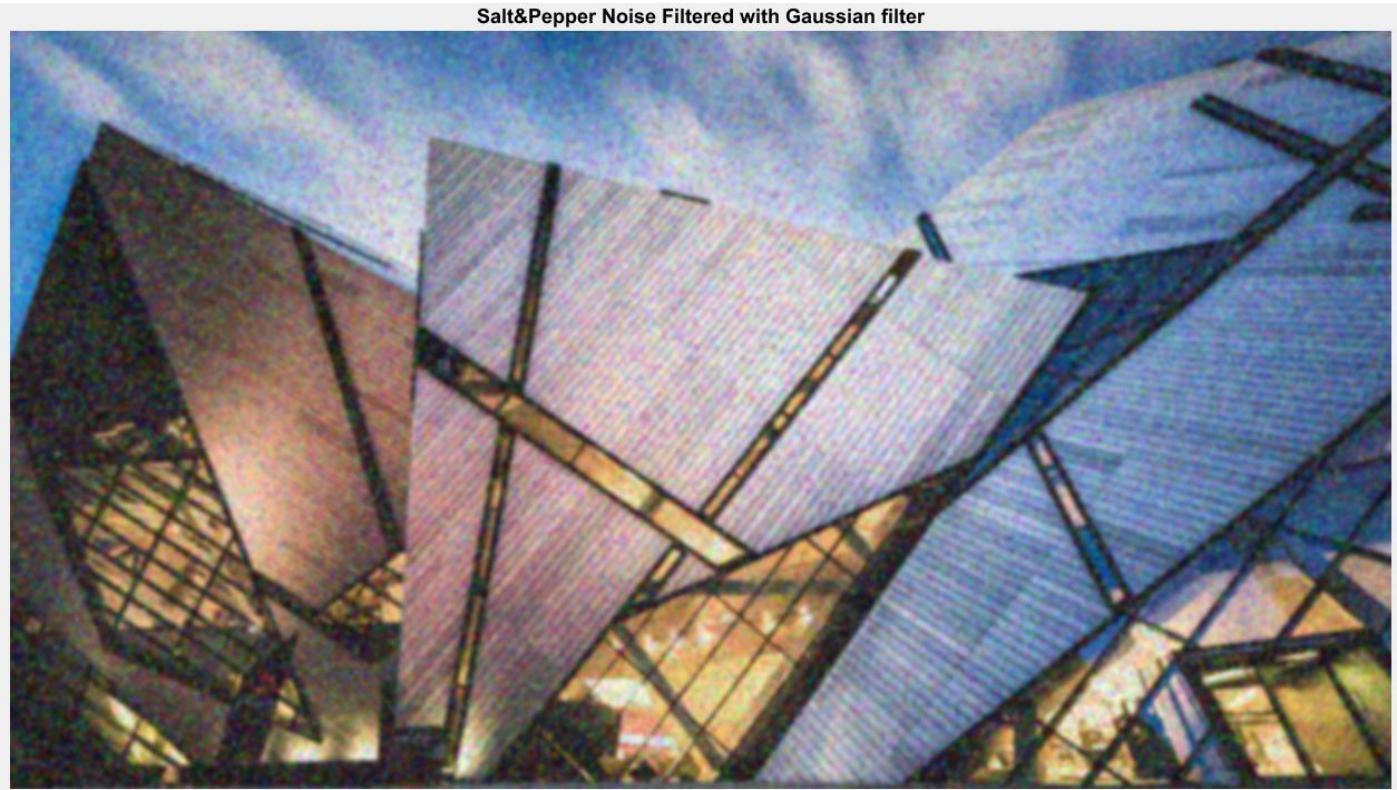
تصویر اول، نویز فلفل-نمکی، فیلتر هموارساز میانگین متحرک

Salt&Pepper Noise Filtered with moving average



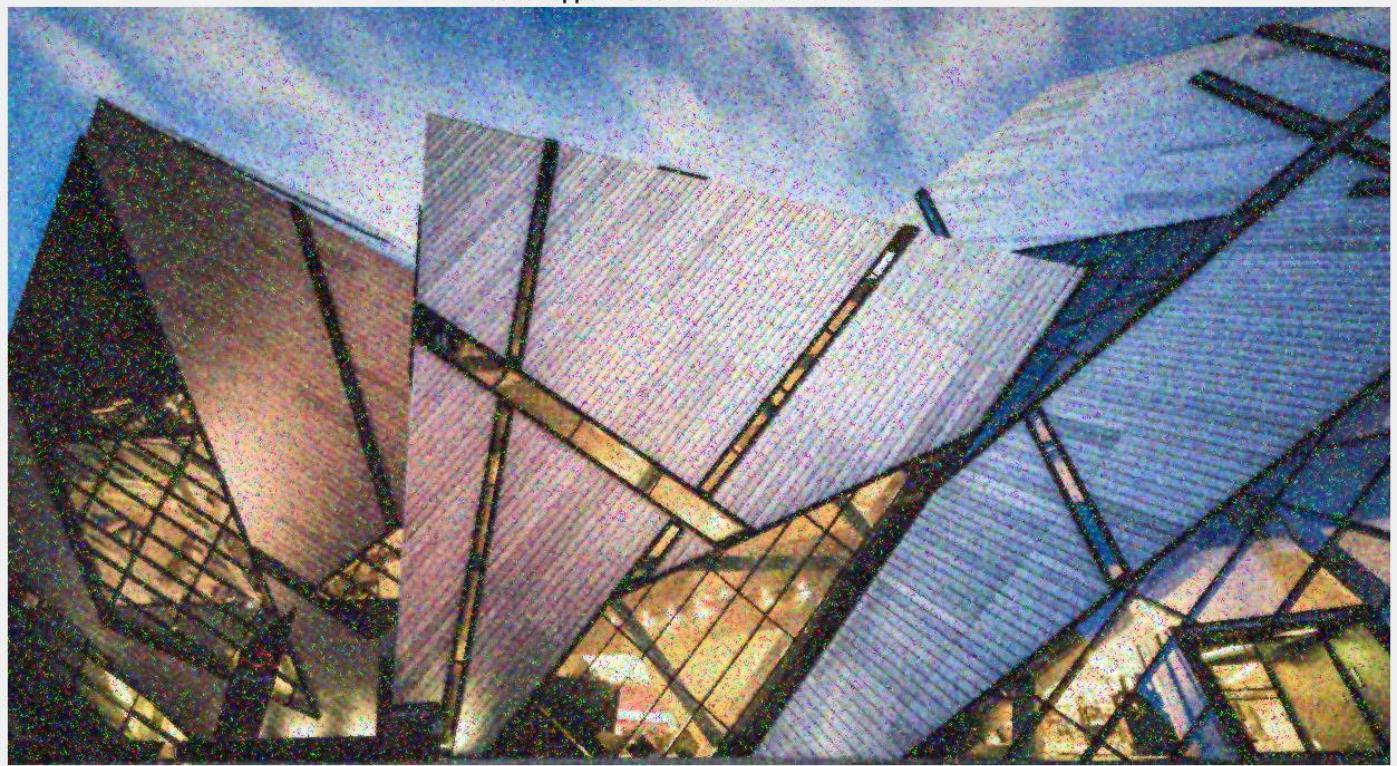
تصویر اول، نویز فلفل-نمکی، فیلتر هموارساز گوسی

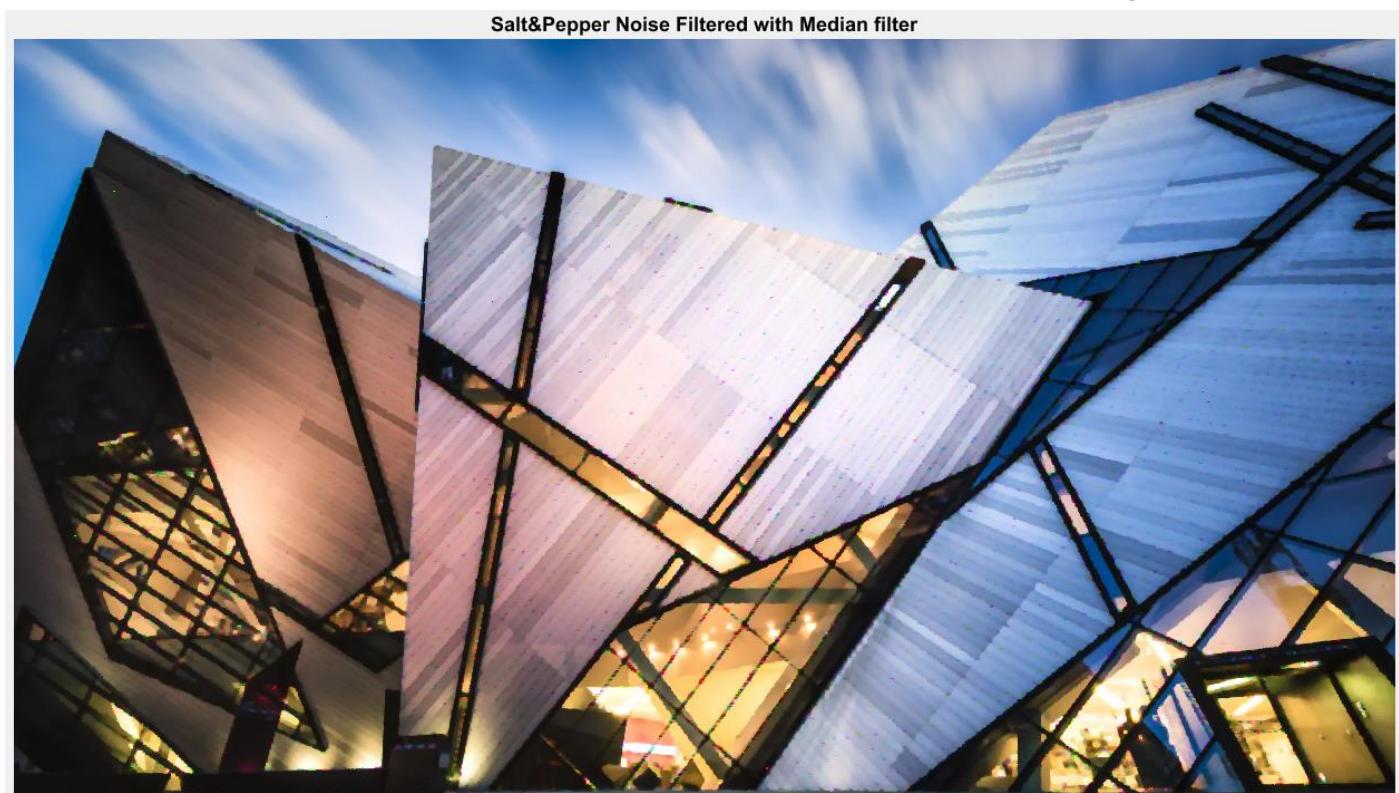
Salt&Pepper Noise Filtered with Gaussian filter



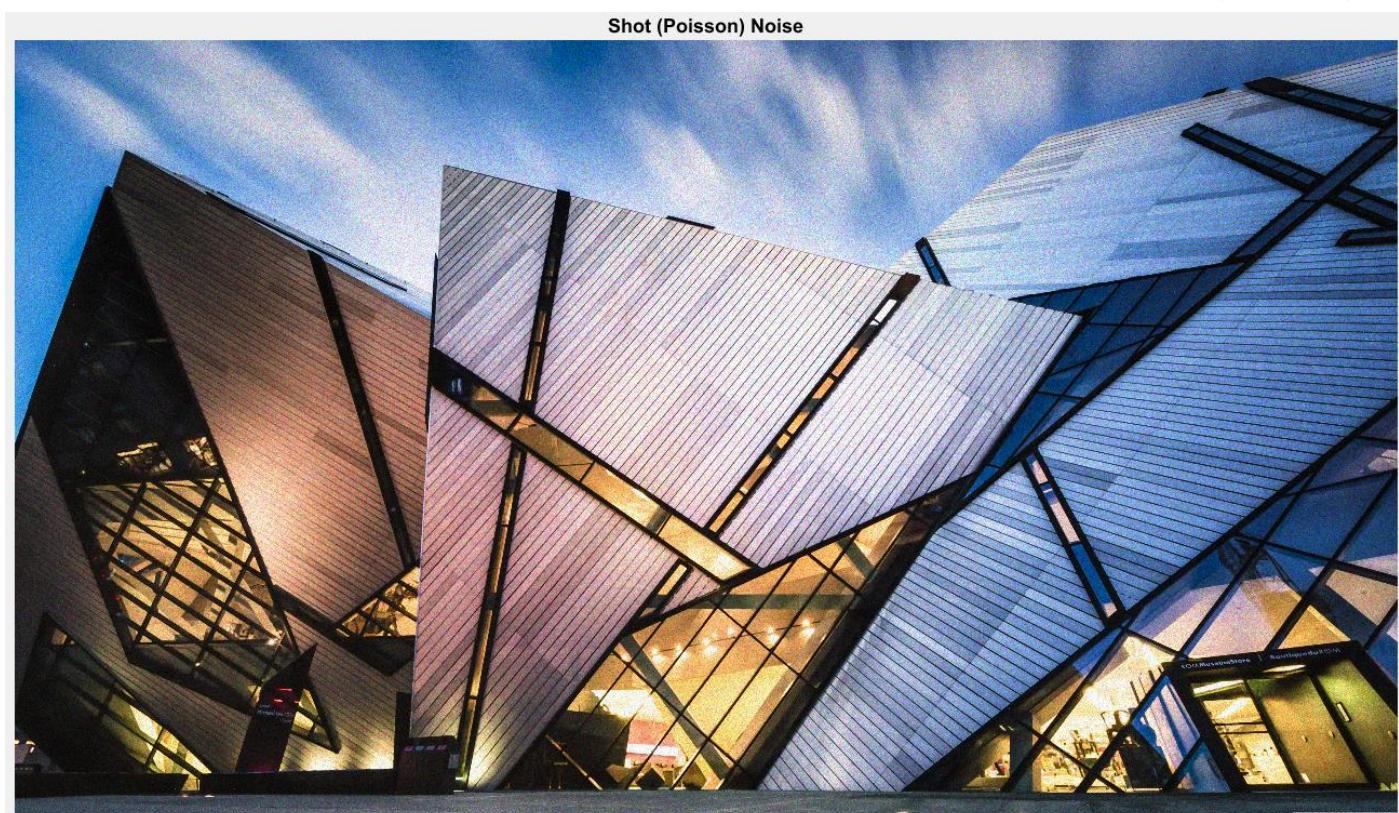
تصویر اول، نویز فلفل-نمکی، فیلتر wiener

Salt&Pepper Noise Filtered with Wiener filter



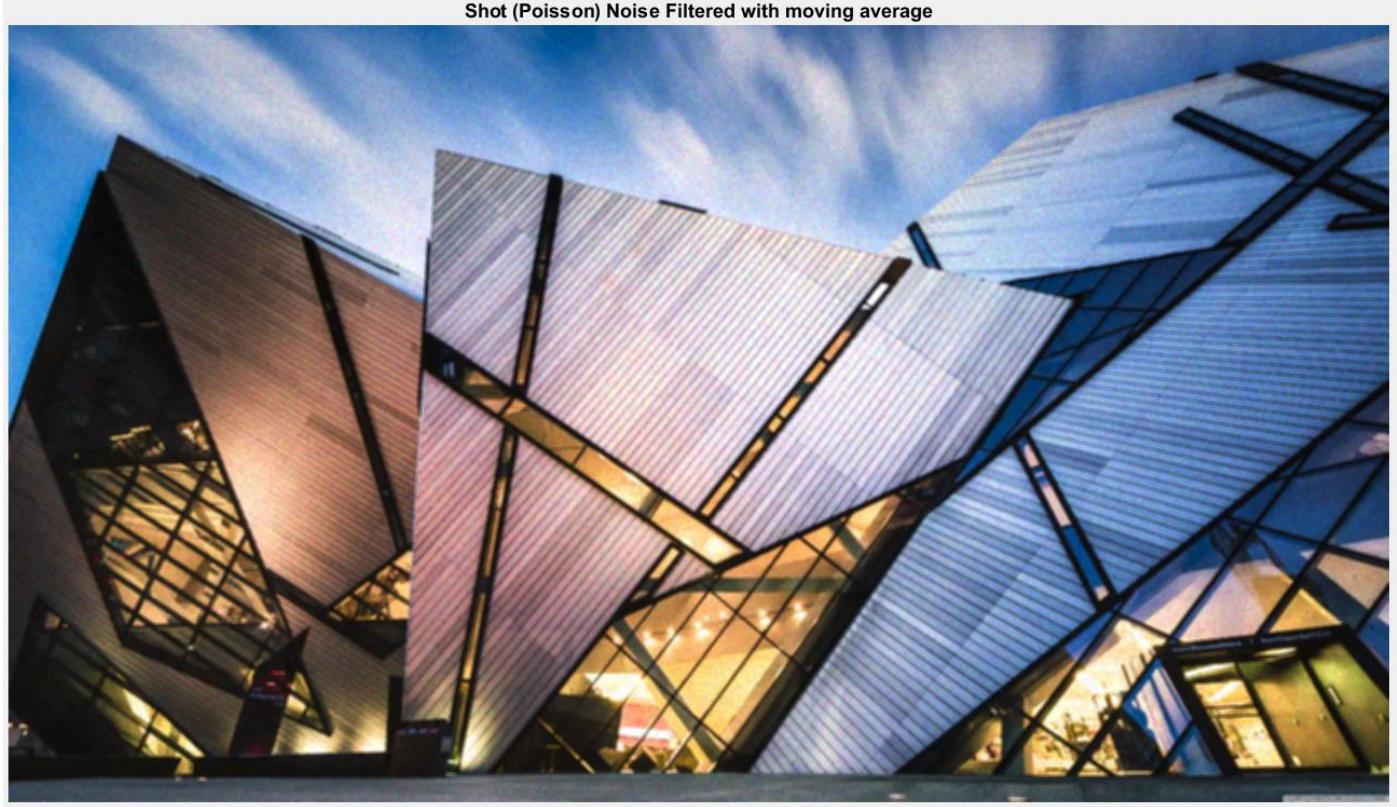


تصویر اول، نویز Shot



تصویر اول، نویز Shot ، فیلتر هموارساز میانگین متحرک

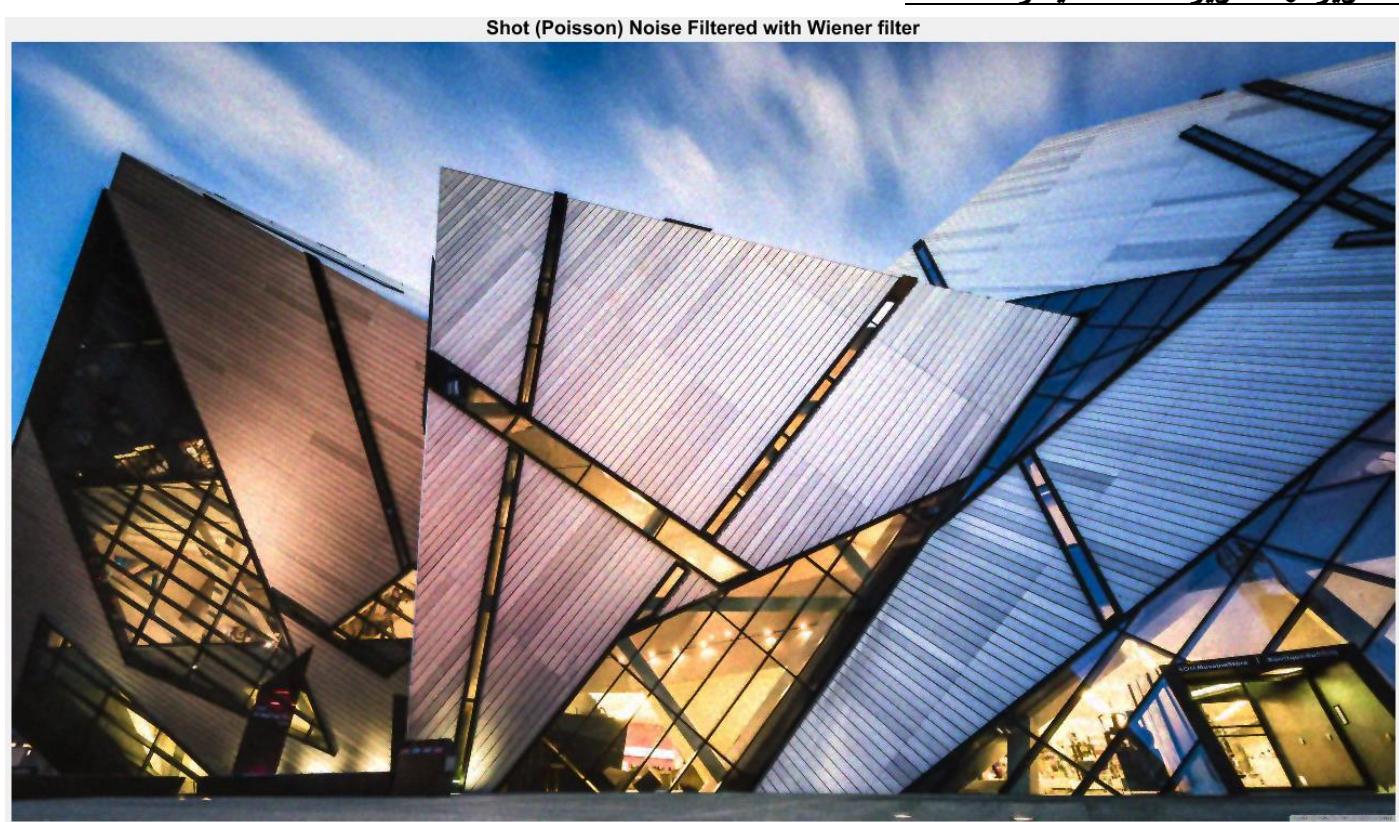
Shot (Poisson) Noise Filtered with moving average



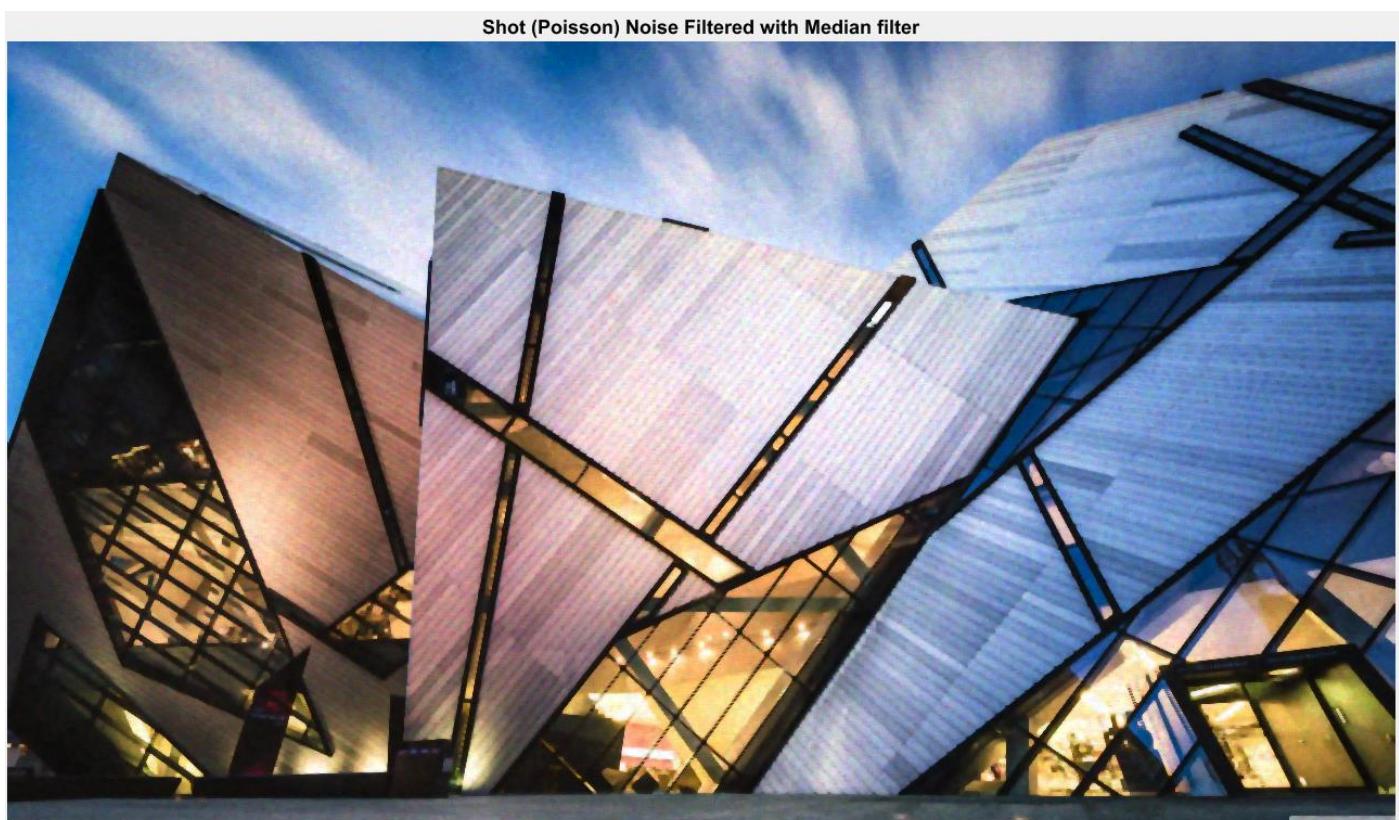
تصویر اول، نویز Shot ، فیلتر هموارساز گوسی

Shot (Poisson) Noise Filtered with Gaussian filter

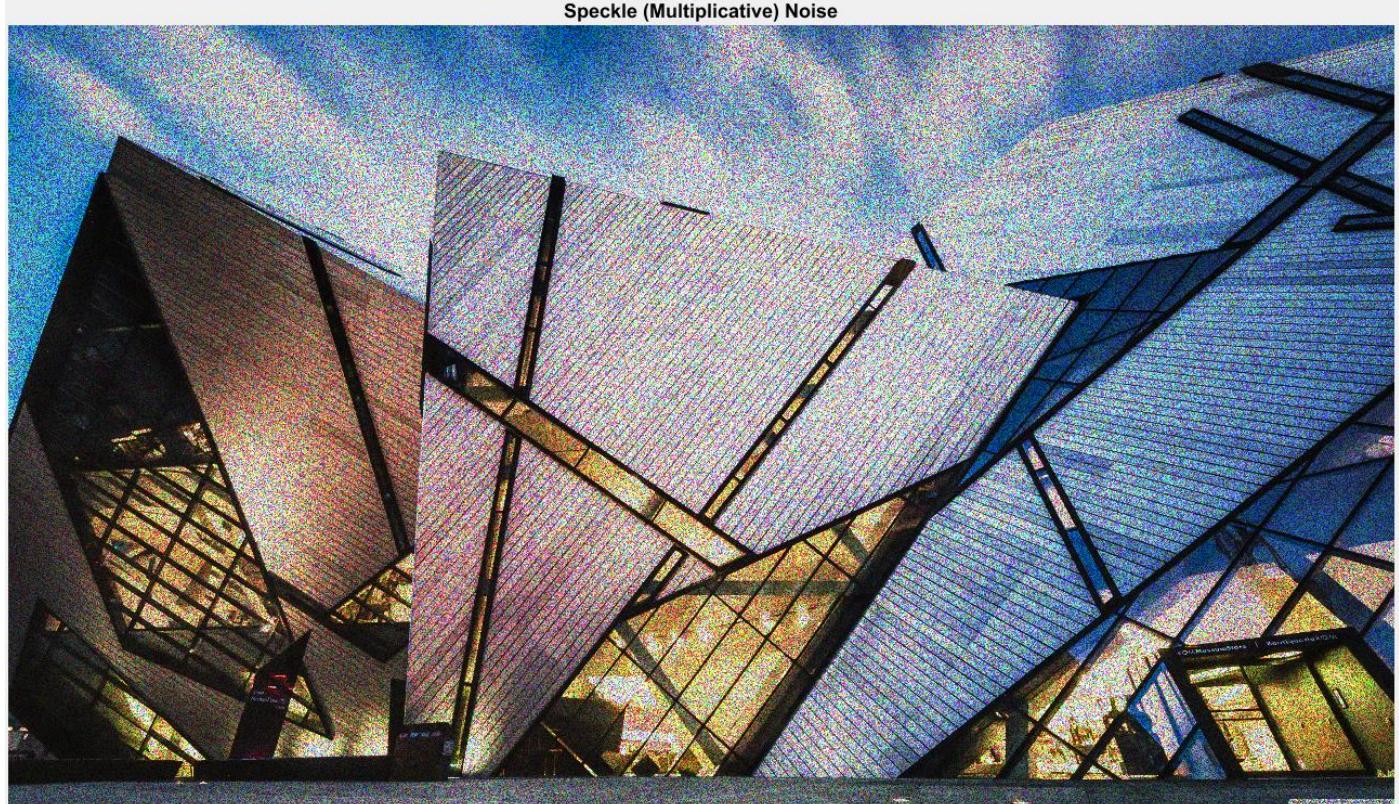




تصویر اول، نویز Shot ، فیلتر میانه

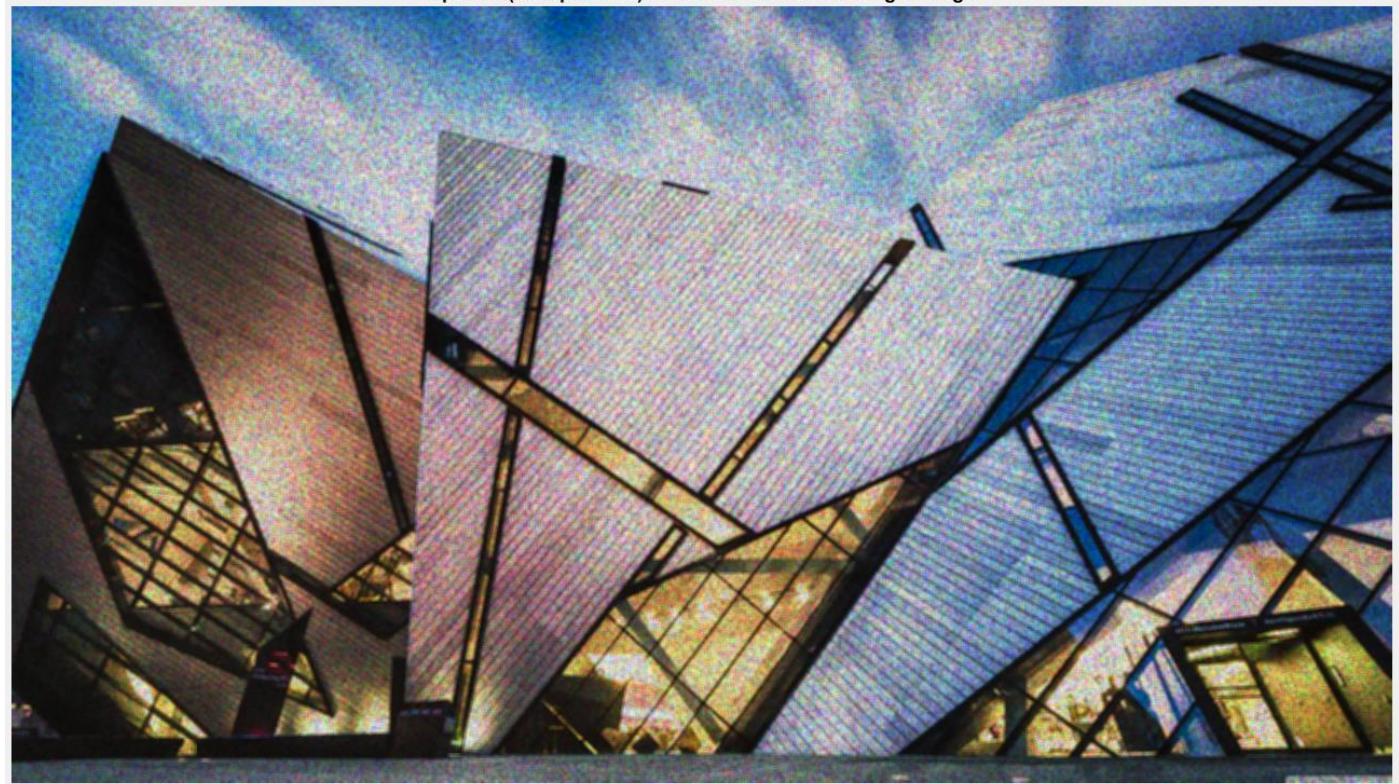


Speckle (Multiplicative) Noise



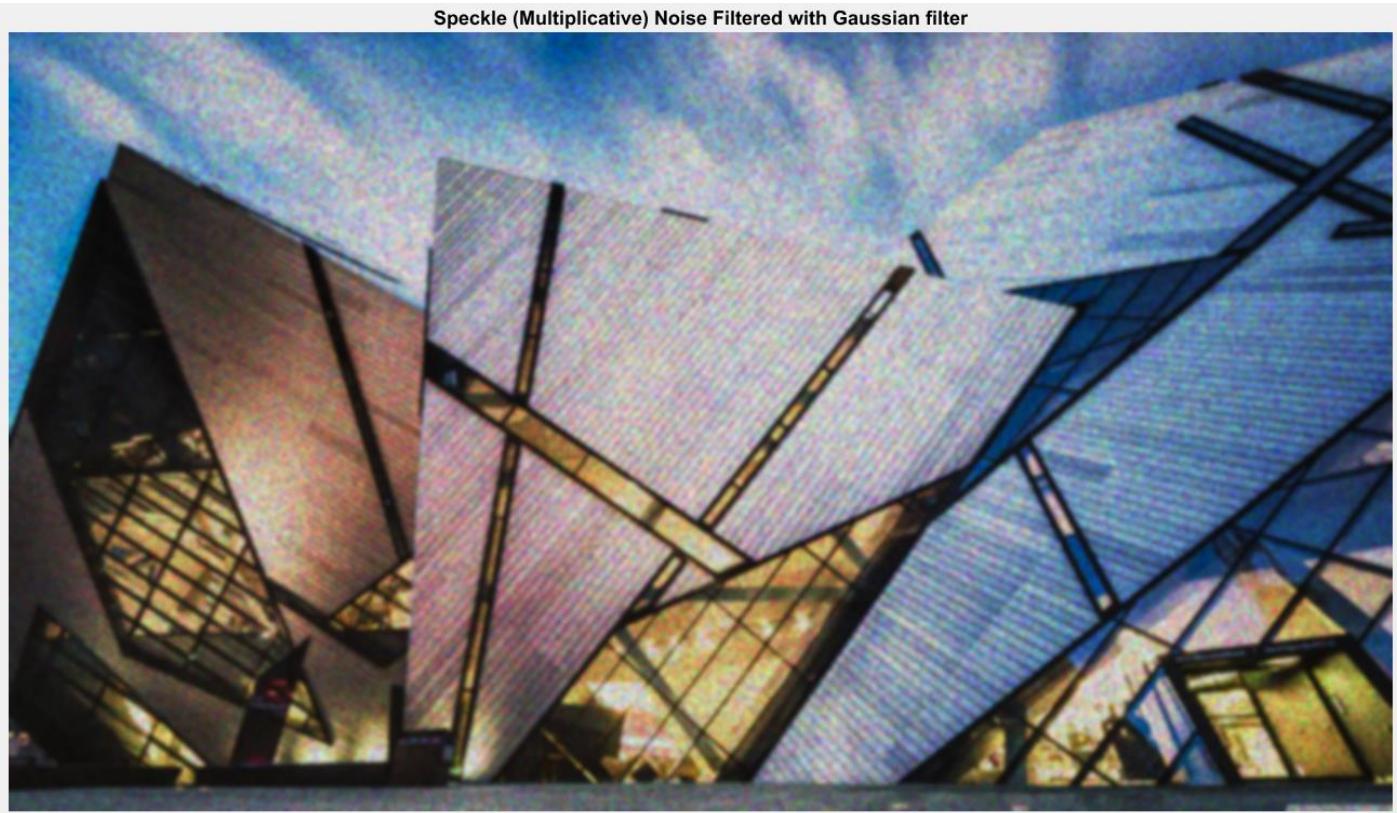
تصویر اول، نویز Speckle (ضرب‌شونده)

Speckle (Multiplicative) Noise Filtered with moving average



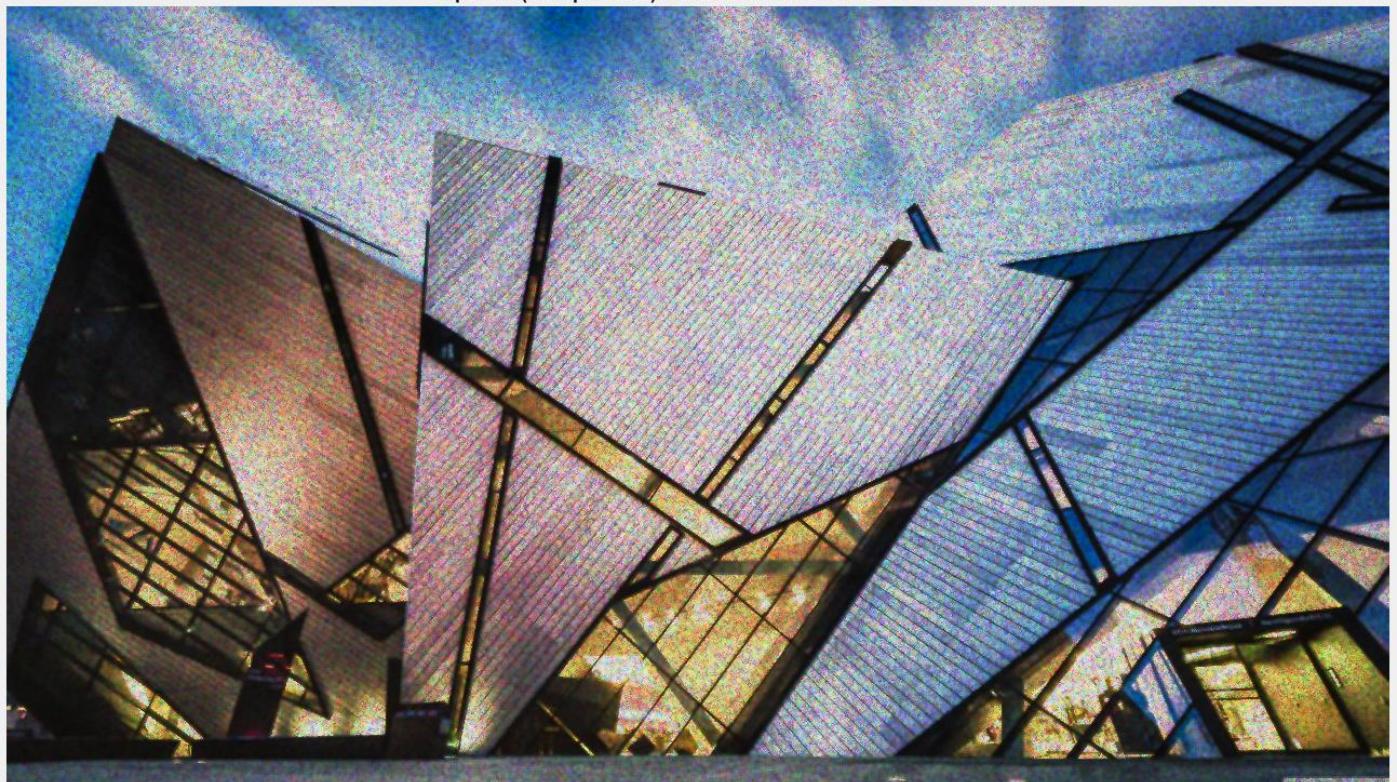
## تصویر اول، نویز Speckle (ضرب‌شونده)، فیلتر هموارساز گوسی

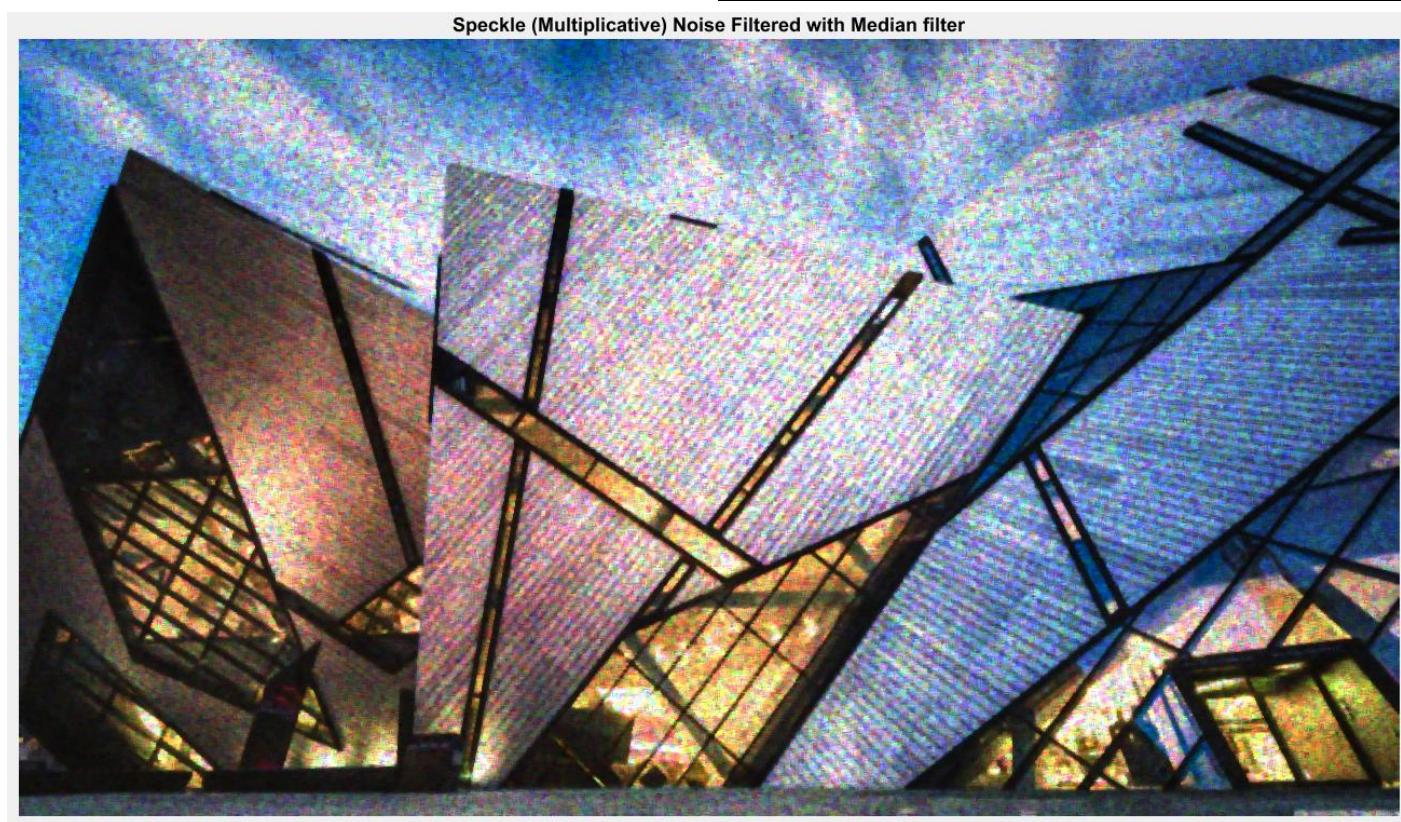
Speckle (Multiplicative) Noise Filtered with Gaussian filter



## تصویر اول، نویز Speckle (ضرب‌شونده)، فیلتر wiener

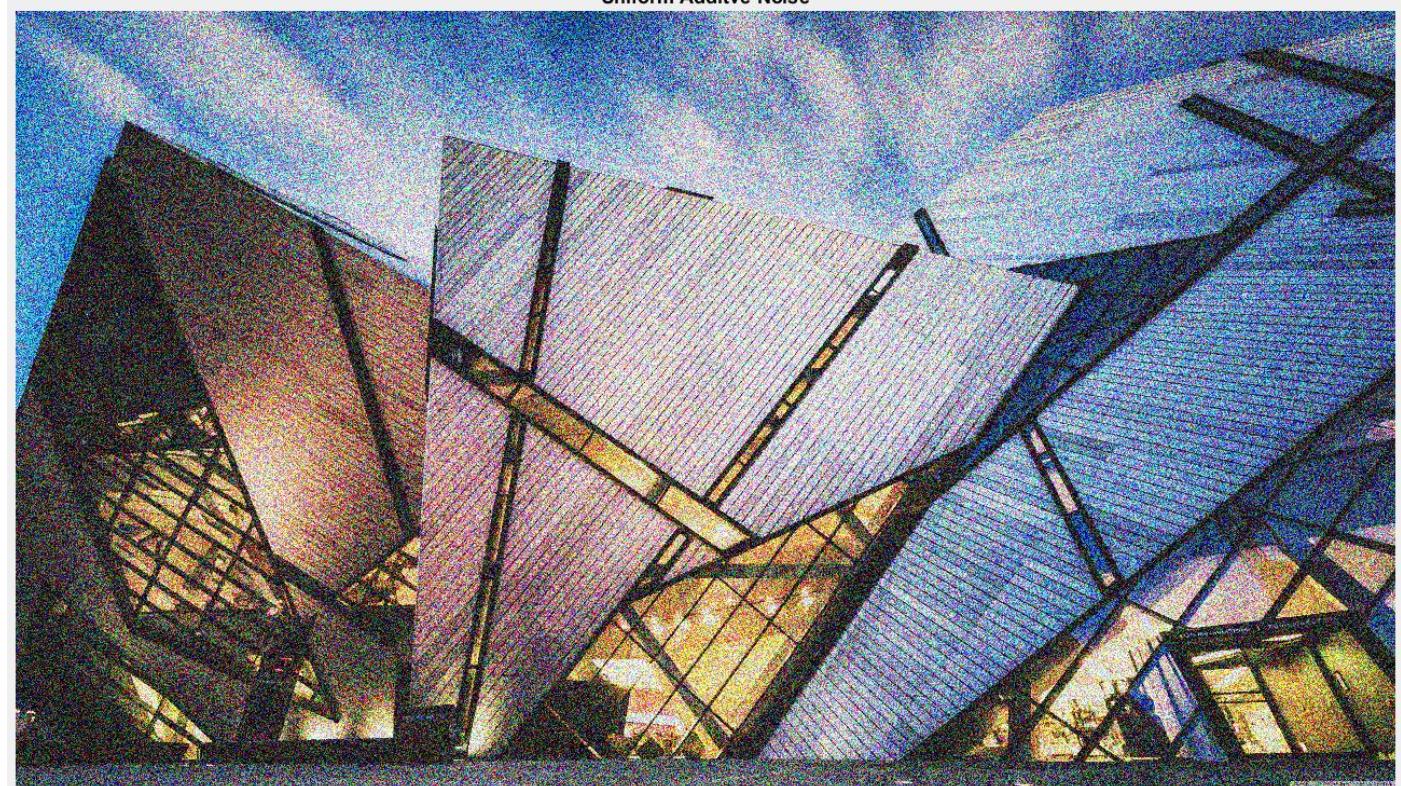
Speckle (Multiplicative) Noise Filtered with Wiener filter





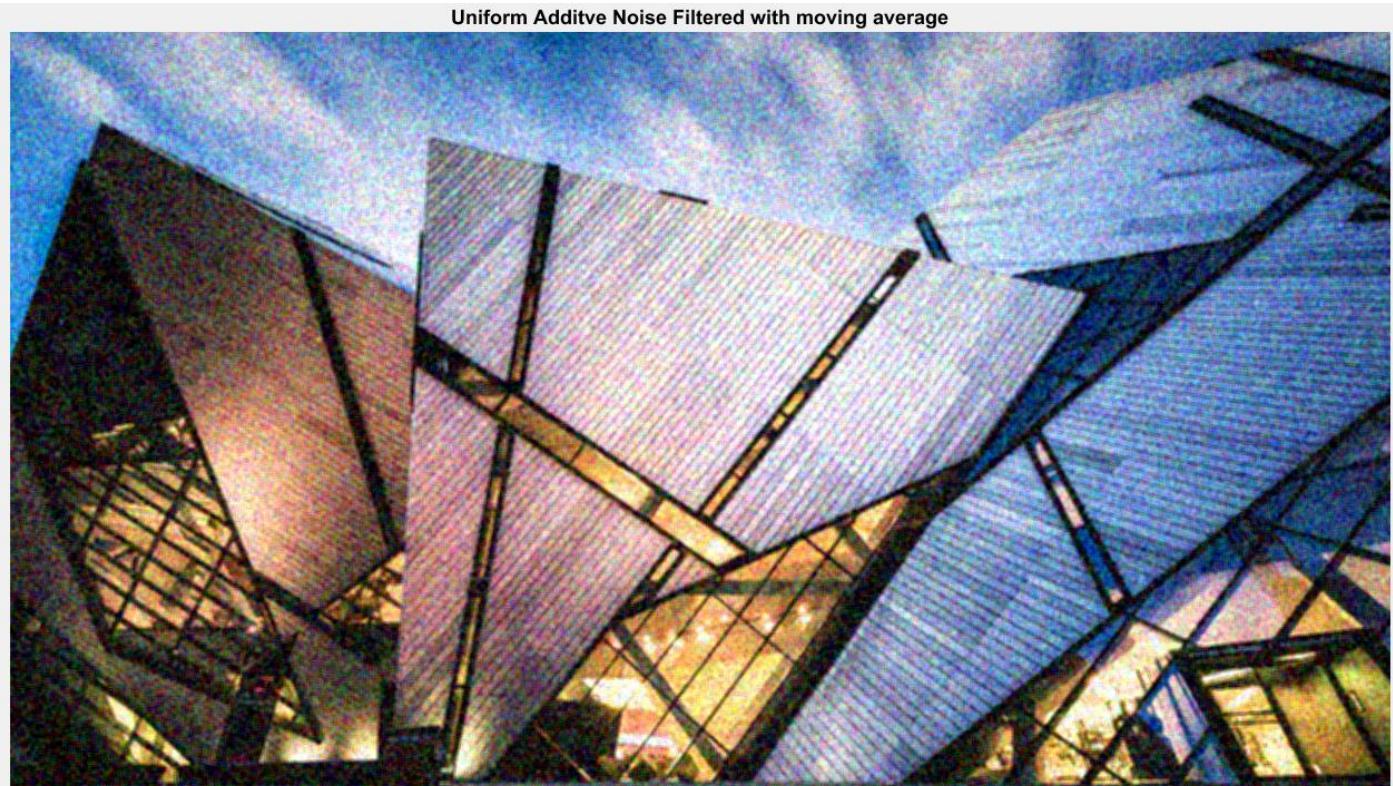
تصویر اول، نویز جمع‌شونده یکنواخت

Uniform Additive Noise



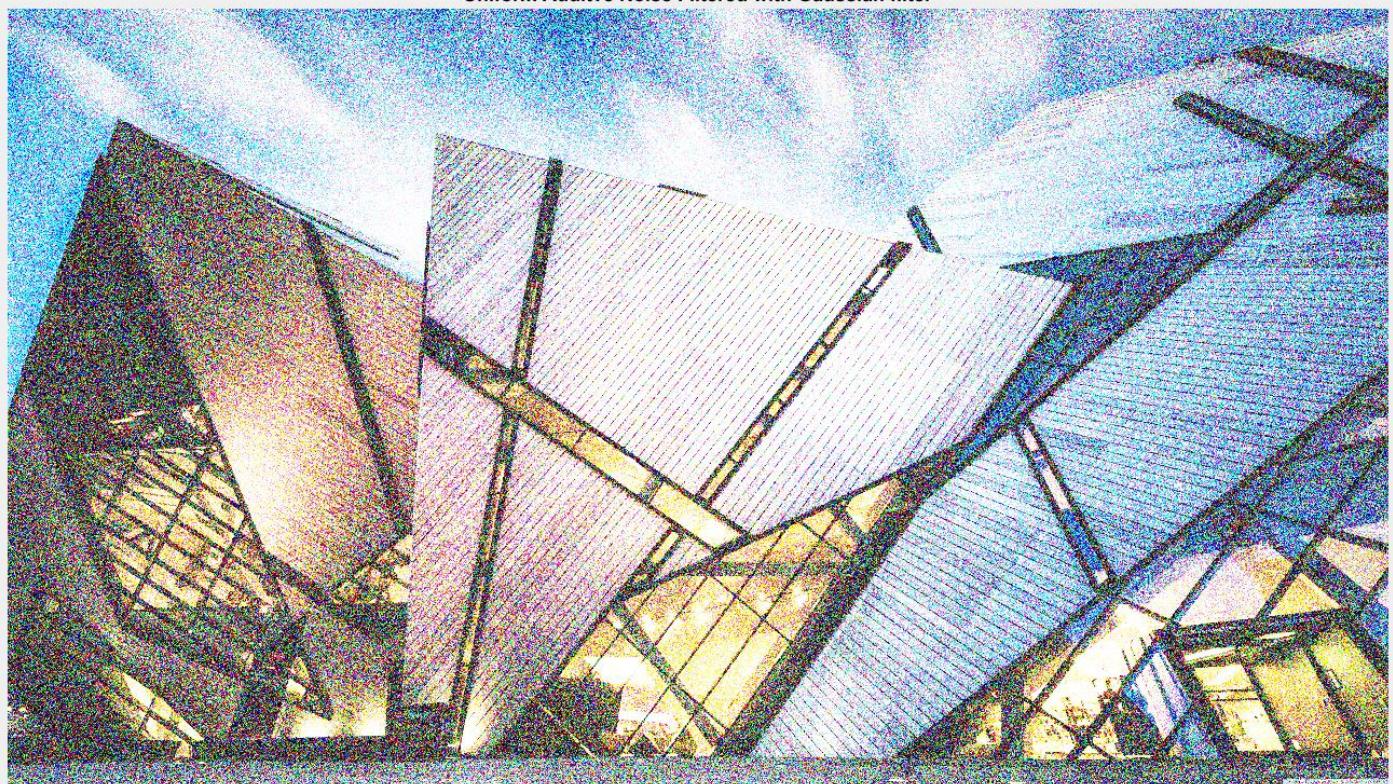
تصویر اول، نویز جمع‌شونده یکنواخت، فیلتر هموارساز میانگین متحرک

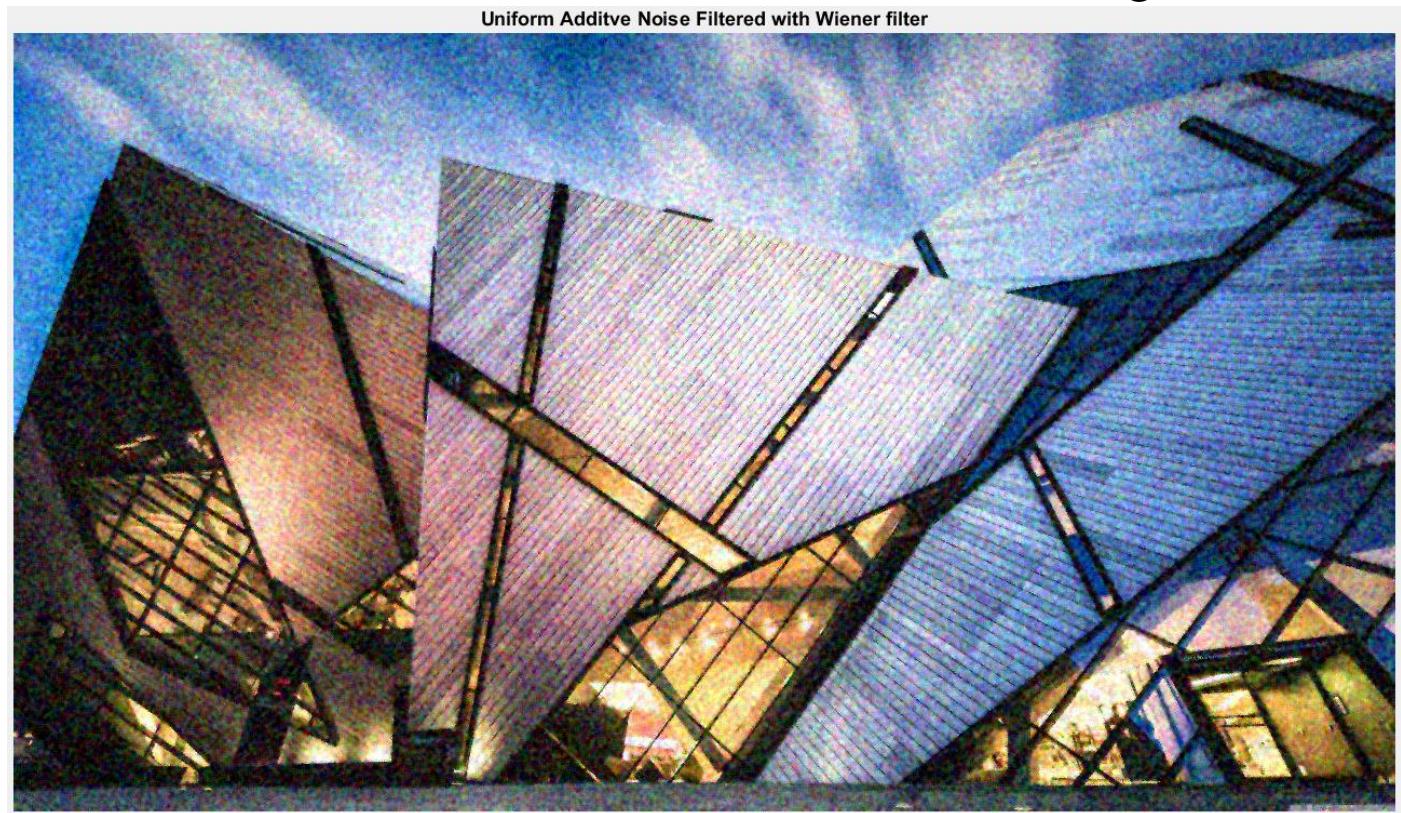
Uniform Additive Noise Filtered with moving average



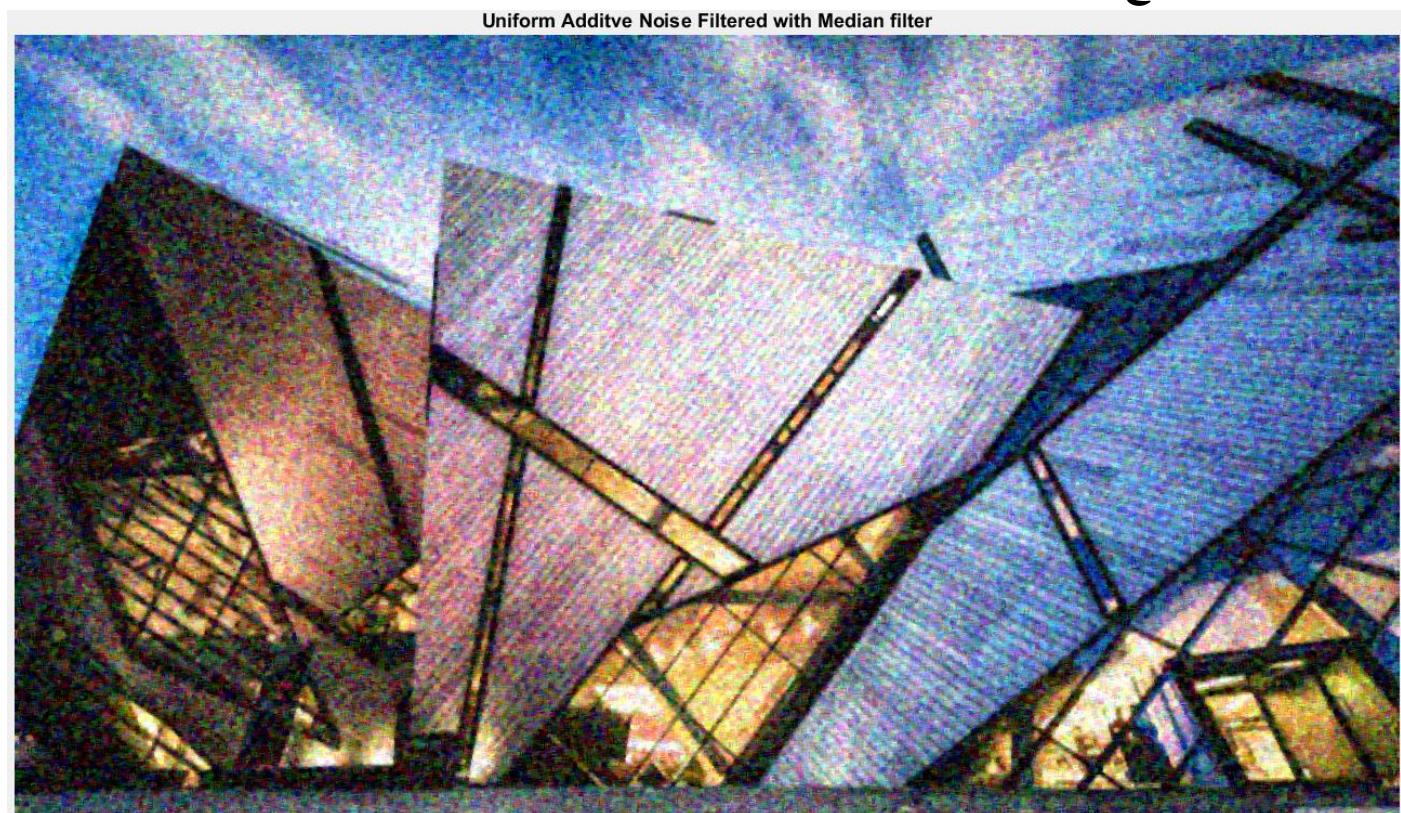
تصویر اول، نویز جمع‌شونده یکنواخت، فیلتر هموارساز گوسی

Uniform Additive Noise Filtered with Gaussian filter





تصویر اول، نویز جمع‌شونده یکنواخت، فیلتر میانه



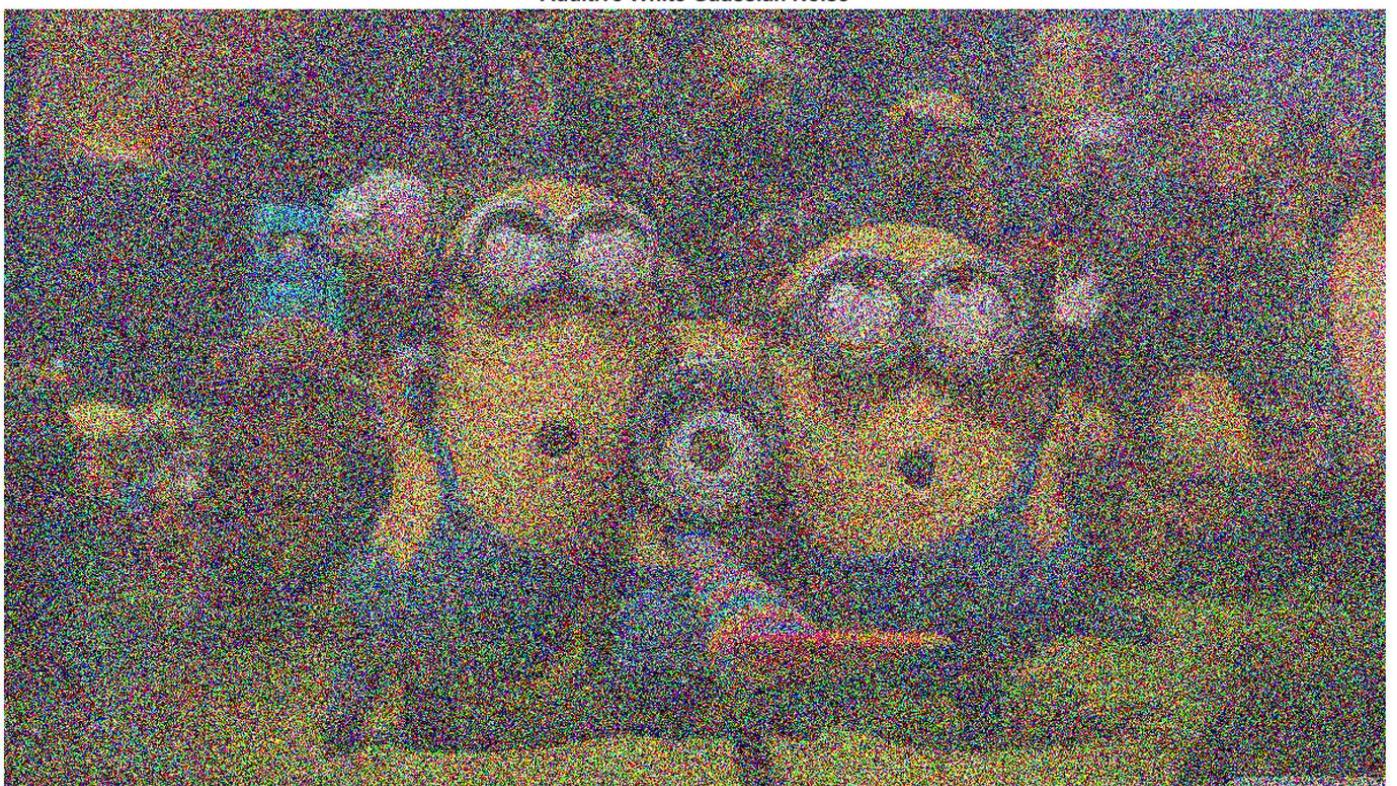
تصویر دوم، بدون نویز

Original Image



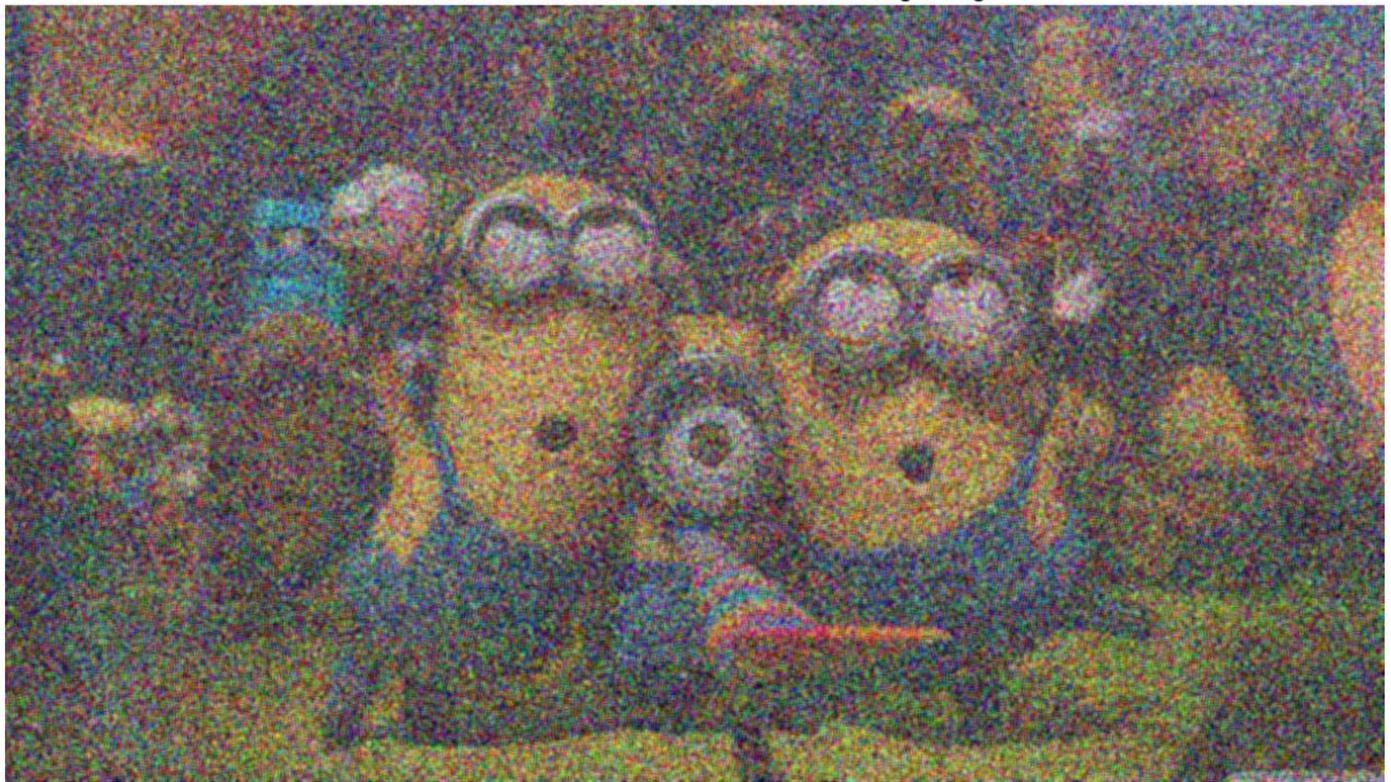
تصویر دوم، نویز جمع‌شونده گوسی

Additive White Gaussian Noise



تصویر دوم، نویز جمع‌شونده گوسی، فیلتر هموارساز میانگین متحرک

Additive White Gaussian Noise Filtered with moving average



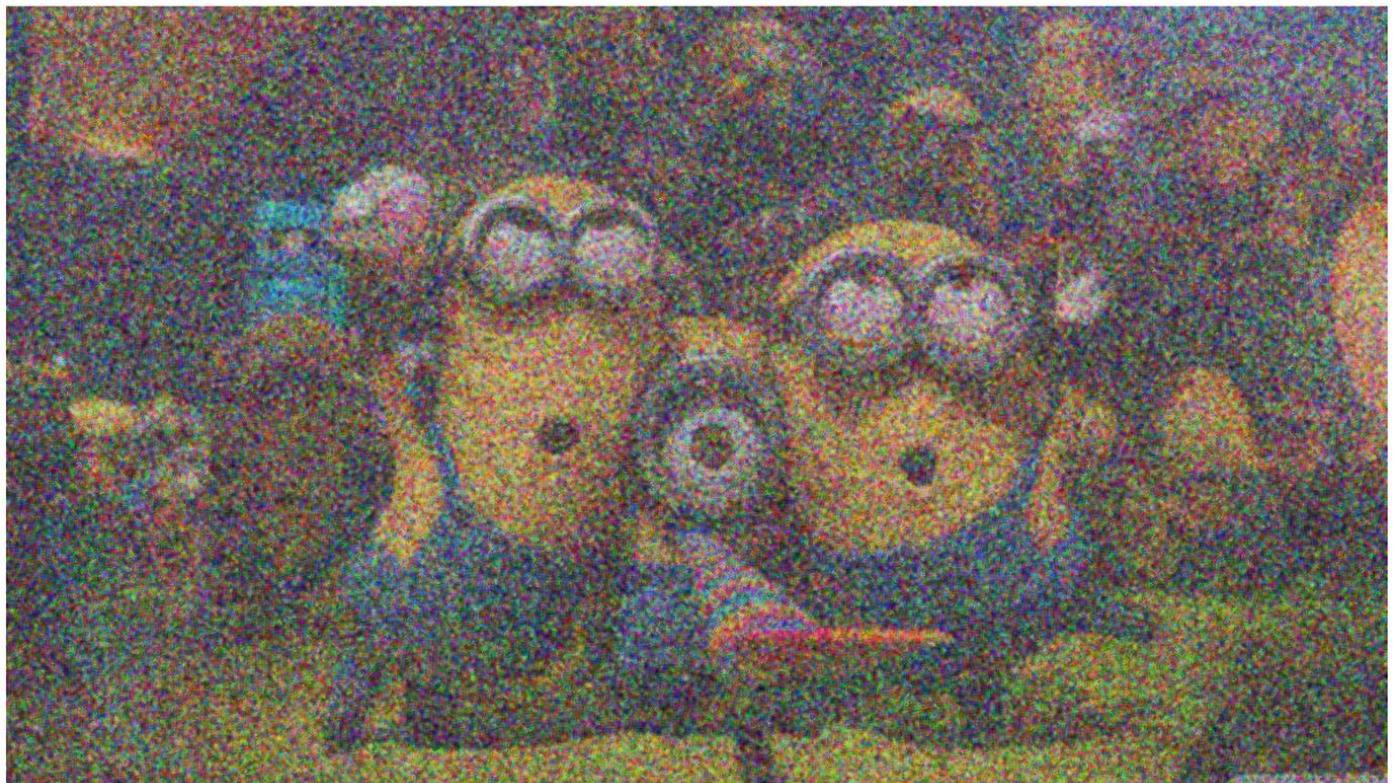
تصویر دوم، نویز جمع‌شونده گوسی، فیلتر هموارساز گوسی

Additive White Gaussian Noise Filtered with Gaussian filter



تصویر دوم، نویز جمع‌شونده گوسی، فیلتر wiener

Additive White Gaussian Noise Filtered with Wiener filter

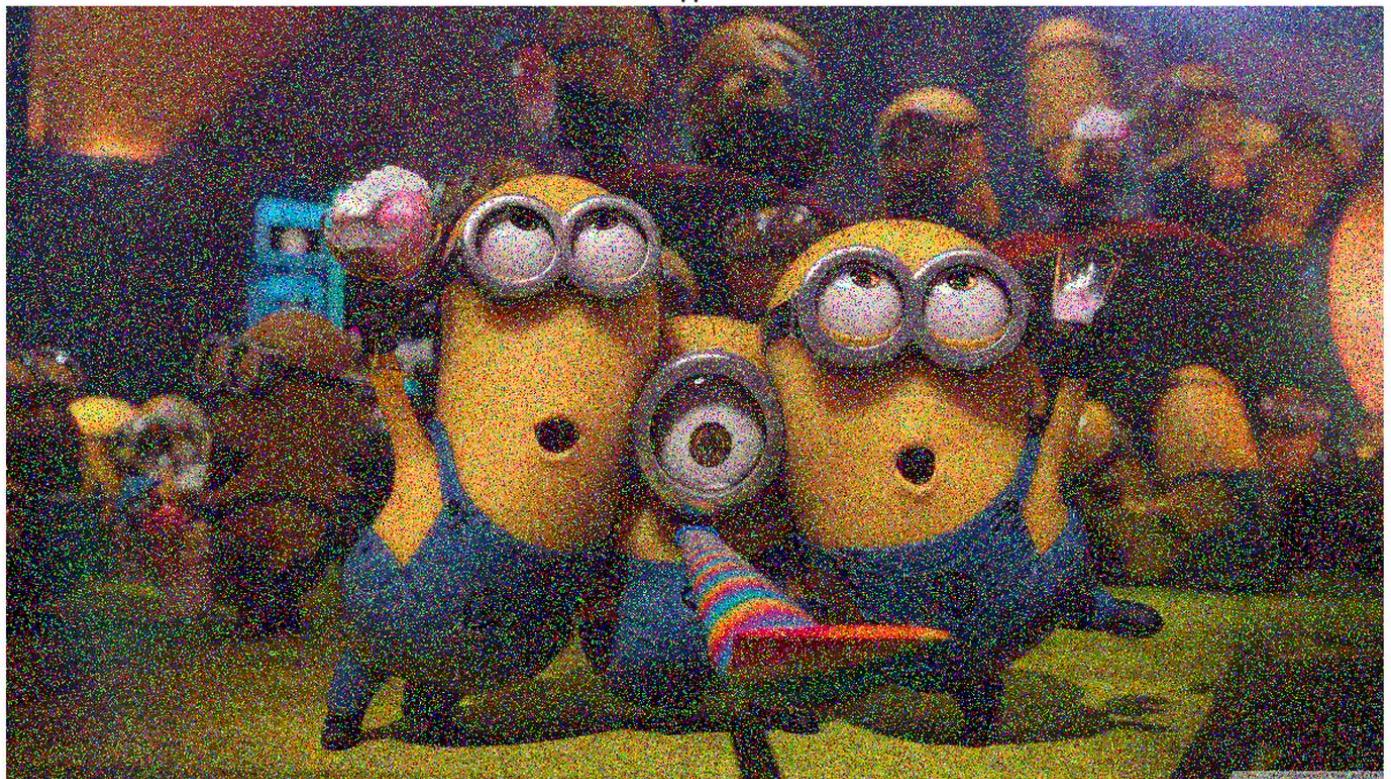


تصویر دوم، نویز جمع‌شونده گوسی، فیلتر میانه

Additive White Gaussian Noise Filtered with Median filter

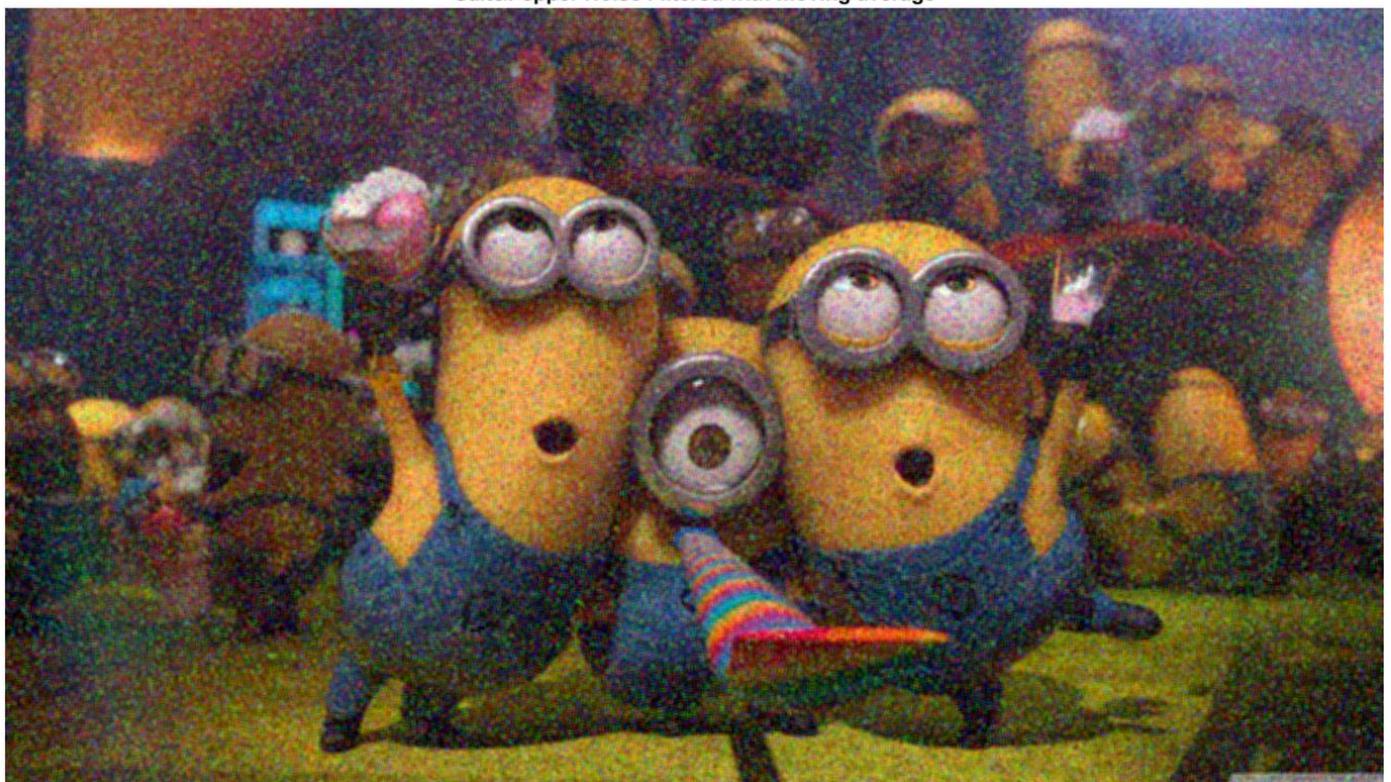


Salt&Pepper Noise



تصویر دوم، نویز فلفل-نمکی، فیلتر هموارساز میانگین متحرک

Salt&Pepper Noise Filtered with moving average



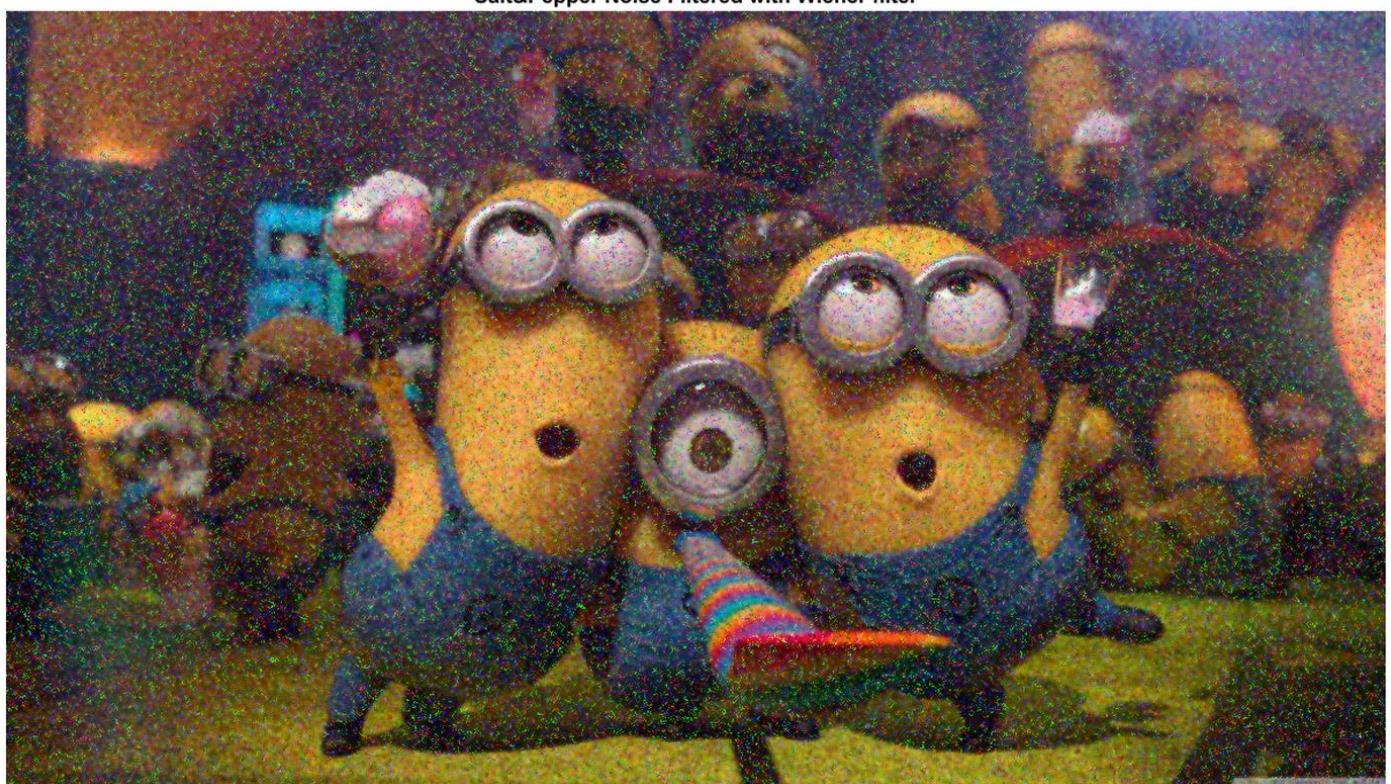
تصویر دوم، نویز فلفل-نمکی، فیلتر هموارساز گوسی

Salt&Pepper Noise Filtered with Gaussian filter



تصویر دوم، نویز فلفل-نمکی، فیلتر wiener

Salt&Pepper Noise Filtered with Wiener filter

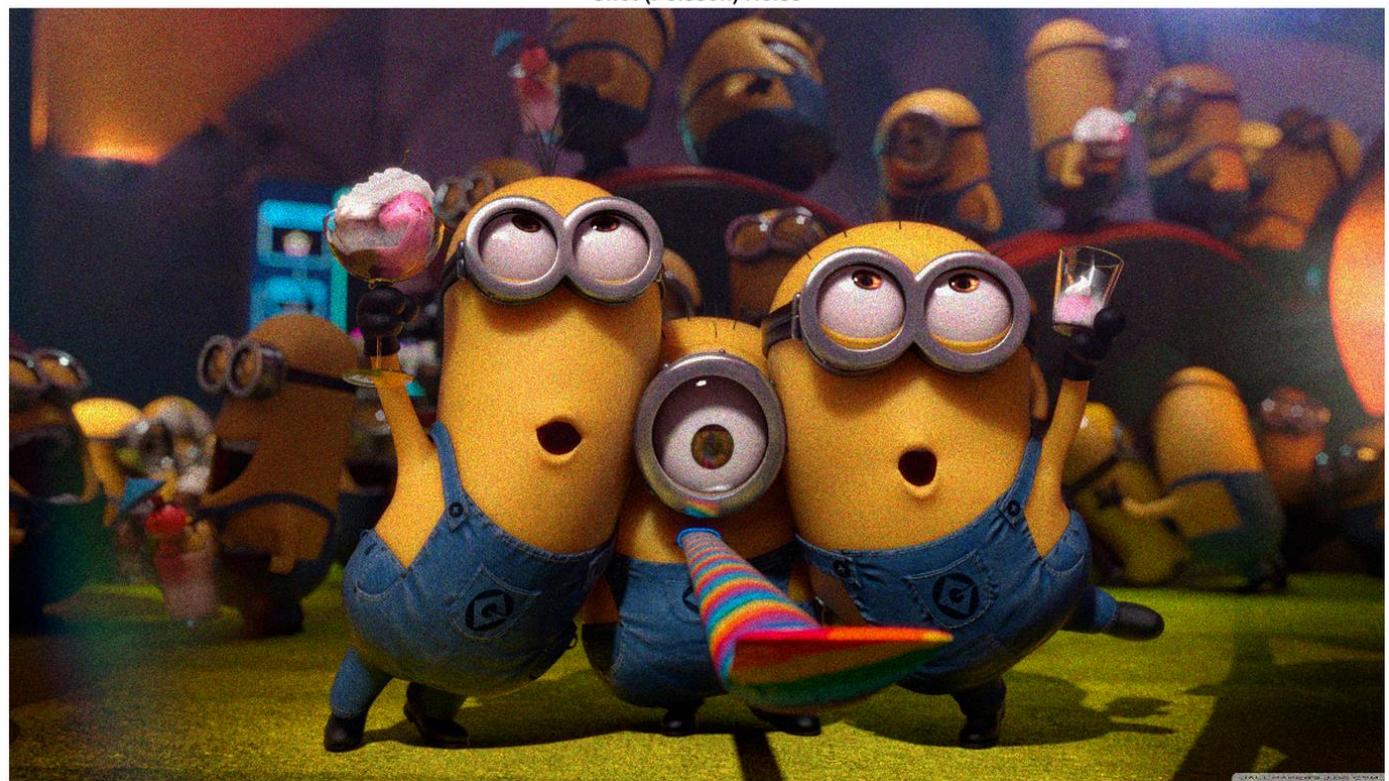


Salt&Pepper Noise Filtered with Median filter



تصویر دوم، نویز Shot

Shot (Poisson) Noise



تصویر دوم، نویز Shot ، فیلتر هموارساز میانگین متحرک

Shot (Poisson) Noise Filtered with moving average



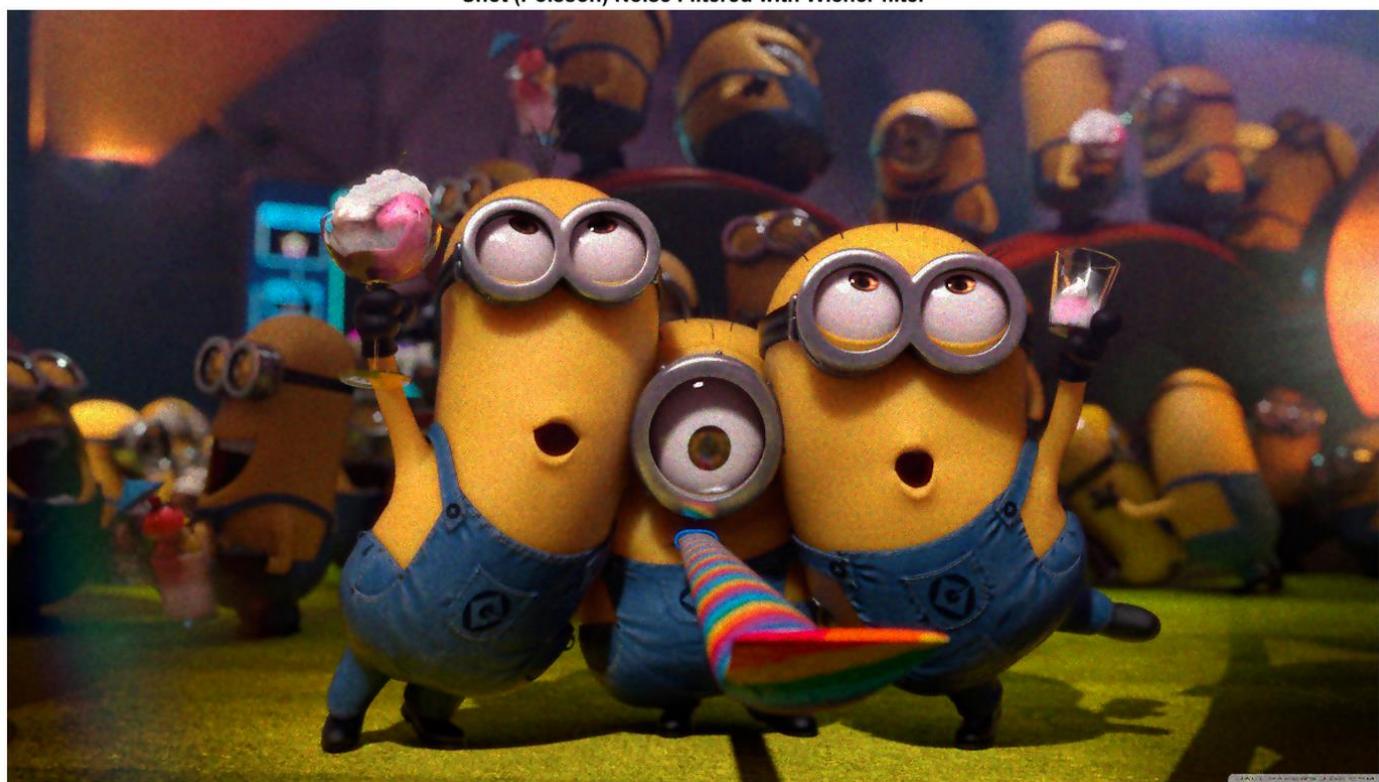
تصویر دوم، نویز Shot ، فیلتر هموارساز گوسی

Shot (Poisson) Noise Filtered with Gaussian filter



تصویر دوم، نویز Shot ، فیلتر wiener

Shot (Poisson) Noise Filtered with Wiener filter

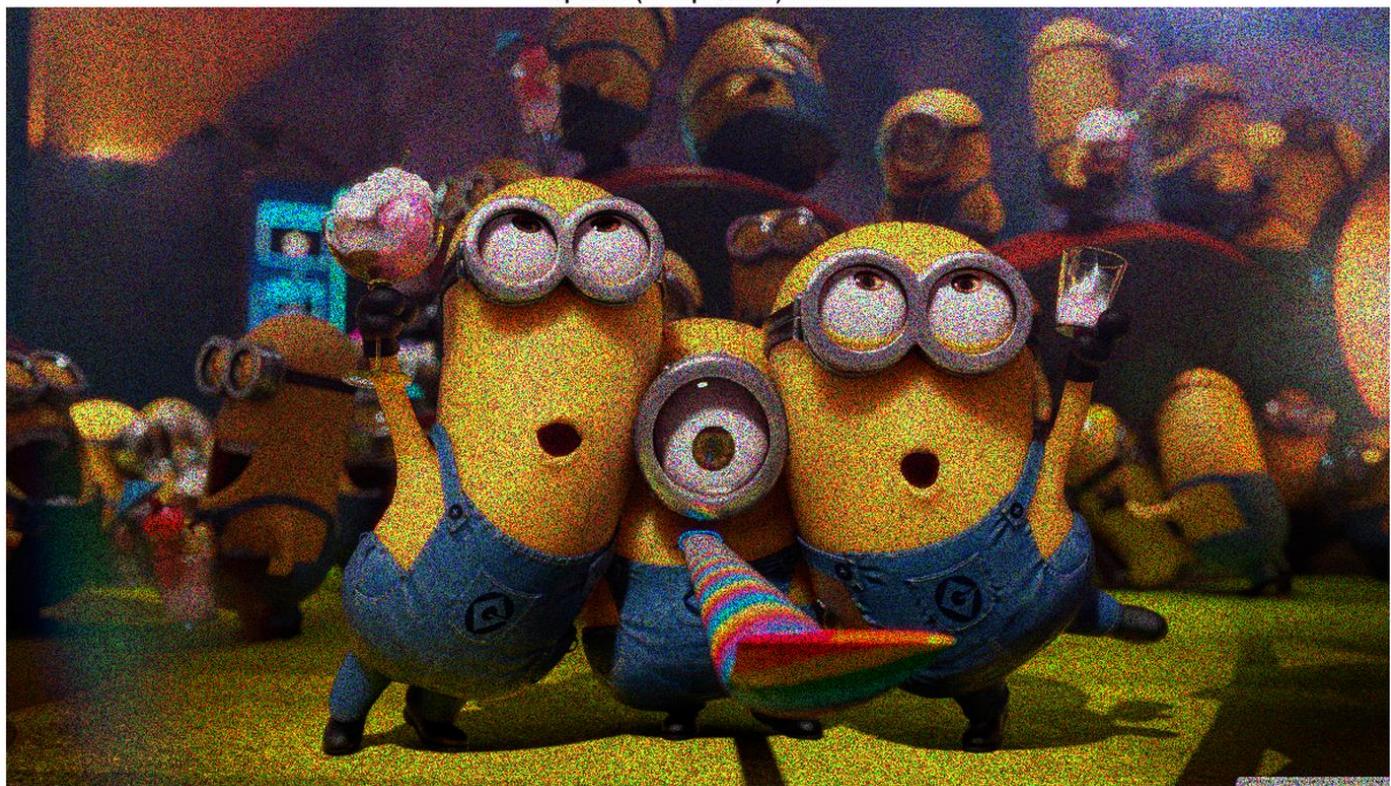


تصویر دوم، نویز Shot ، فیلتر میانه

Shot (Poisson) Noise Filtered with Median filter



Speckle (Multiplicative) Noise



تصویر دوم، نویز Speckle (ضرب‌شونده)، فیلتر هموارساز میانگین متحرک

Speckle (Multiplicative) Noise Filtered with moving average



## تصویر دوم، نویز Speckle (ضرب‌شونده)، فیلتر هموارساز گوسی

Speckle (Multiplicative) Noise Filtered with Gaussian filter



## تصویر دوم، نویز Speckle (ضرب‌شونده)، فیلتر wiener

Speckle (Multiplicative) Noise Filtered with Wiener filter



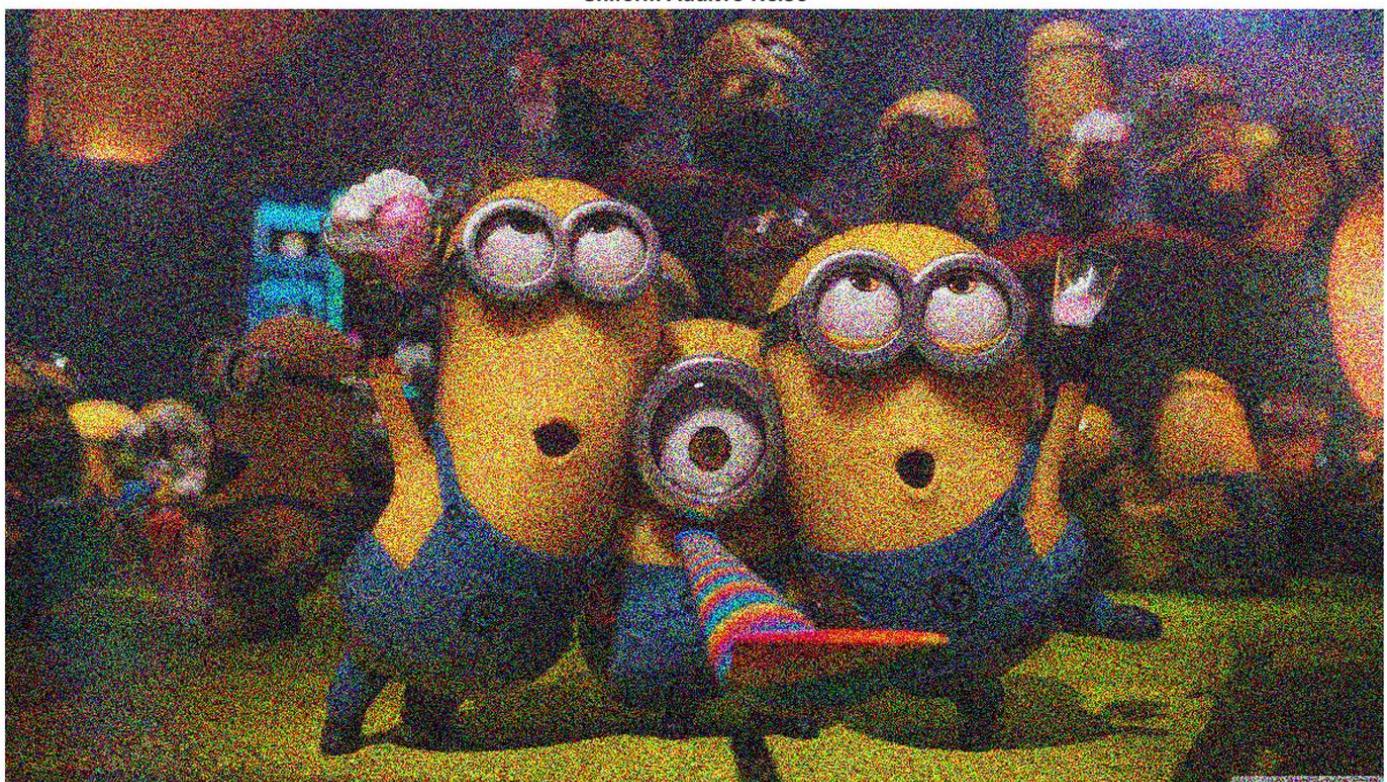
تصویر دوم، نویز Speckle (ضرب‌شونده)، فیلتر میانه

Speckle (Multiplicative) Noise Filtered with Median filter



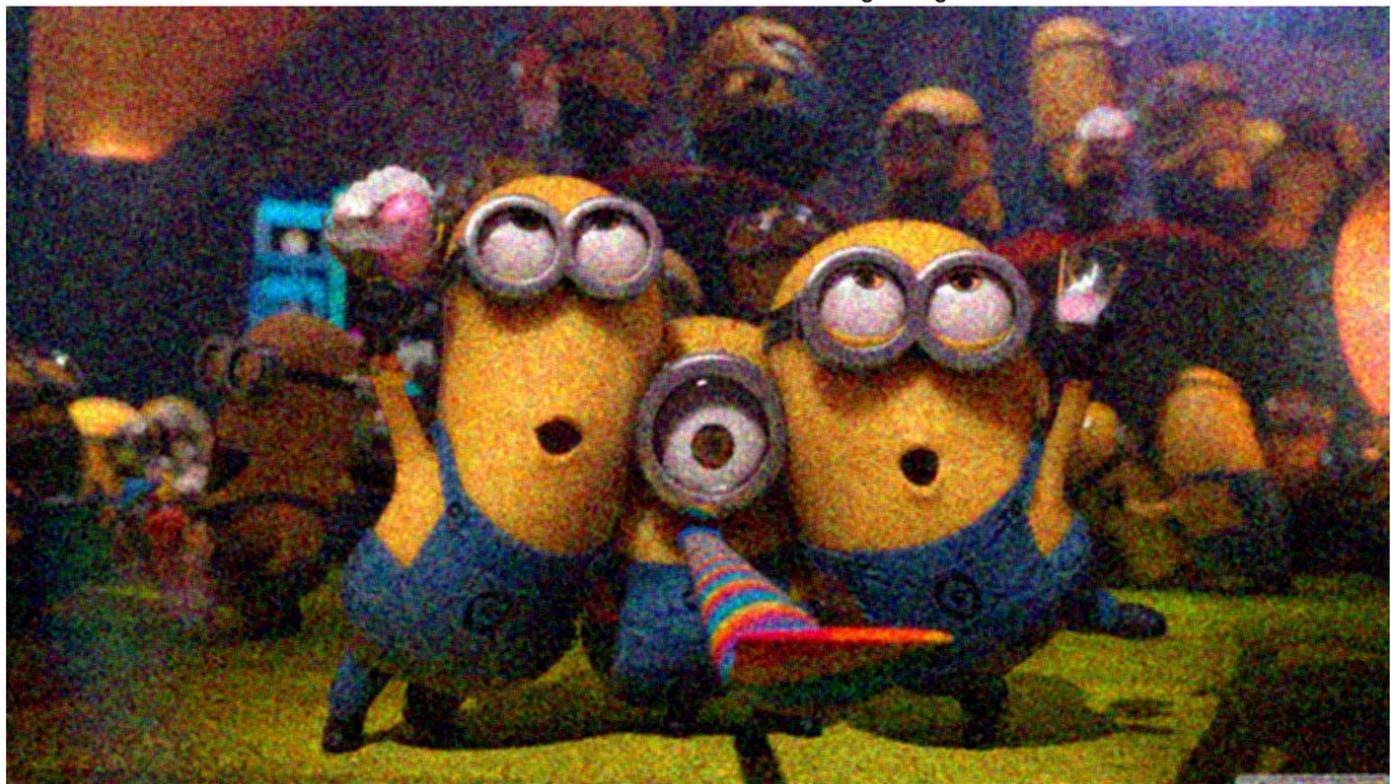
تصویر دوم، نویز جمع‌شونده یکنواخت

Uniform Additive Noise



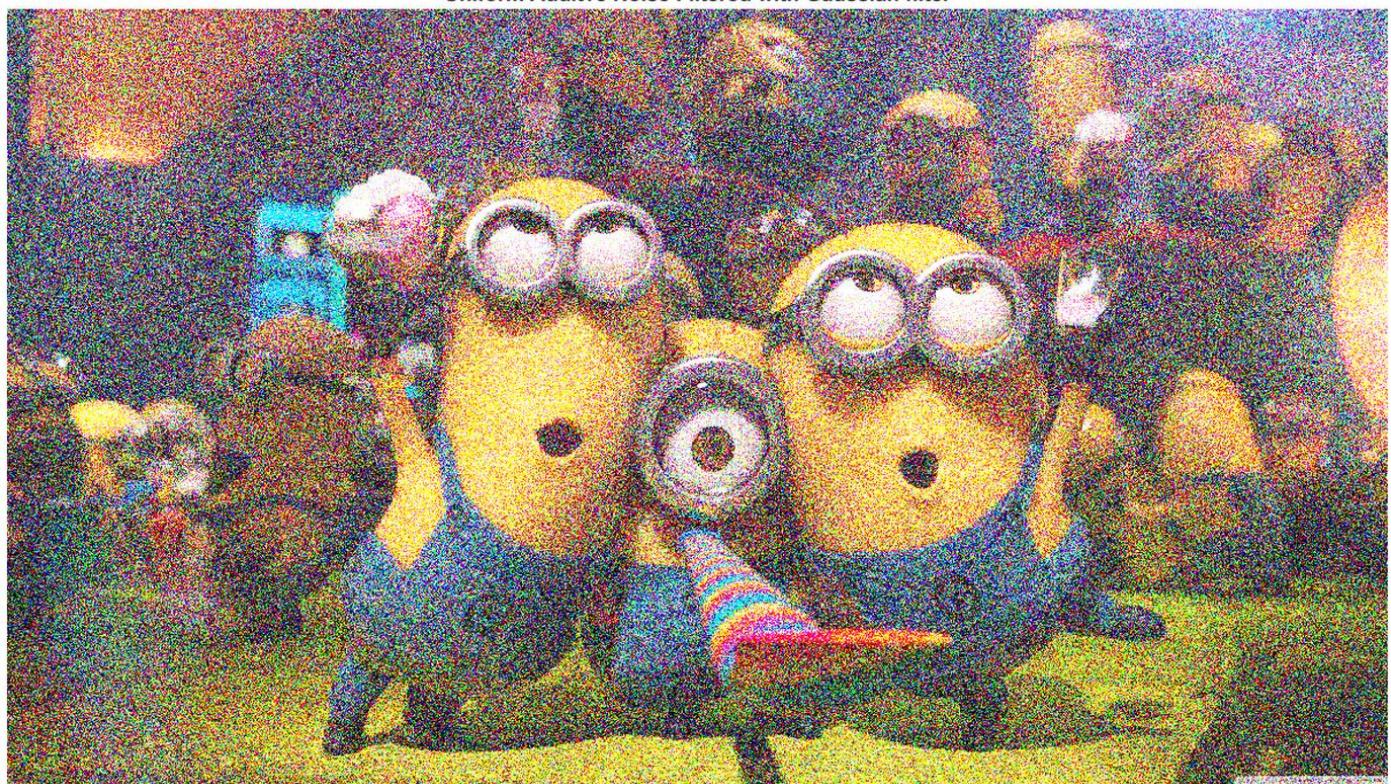
تصویر دوم، نویز جمع‌شونده یکنواخت، فیلتر هموارساز میانگین متحرک

Uniform Additive Noise Filtered with moving average



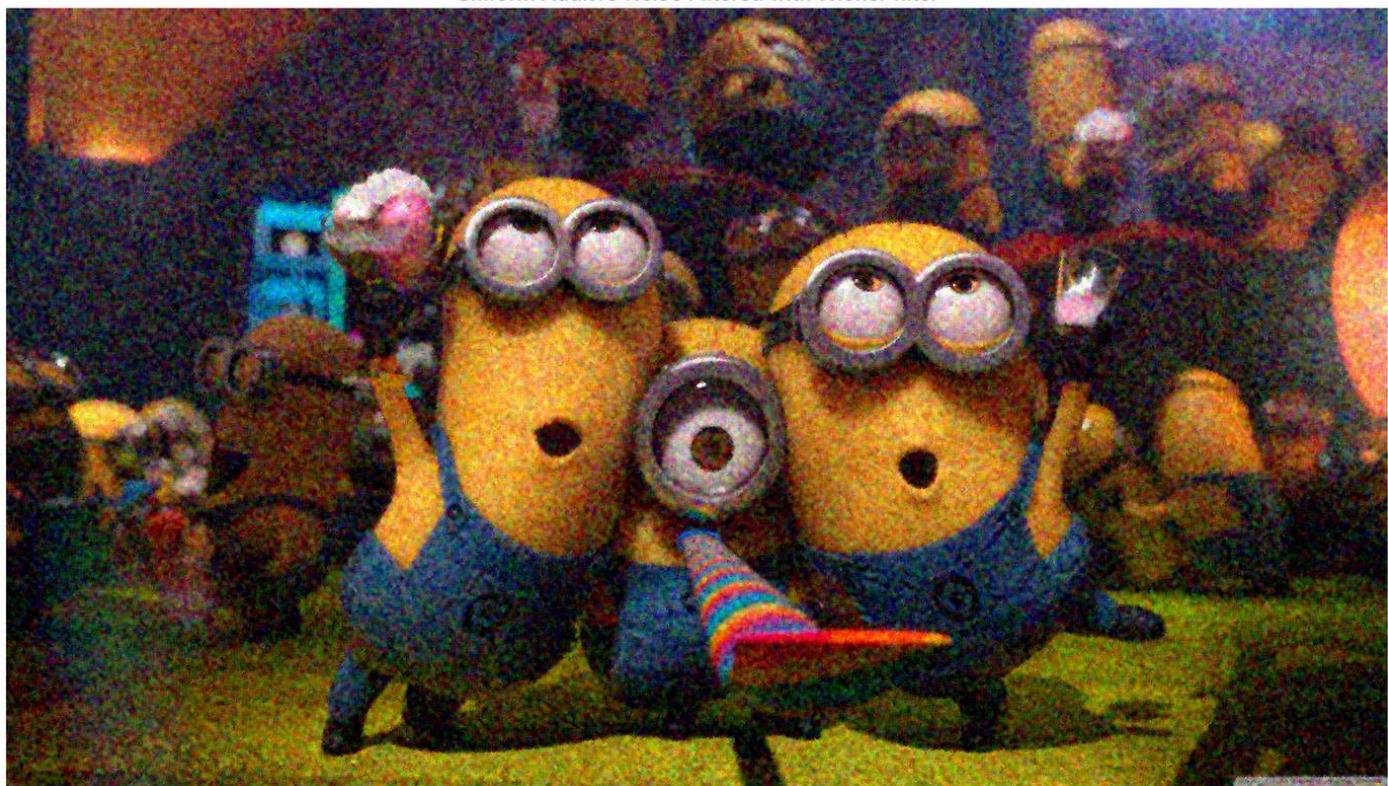
تصویر دوم، نویز جمع‌شونده یکنواخت، فیلتر هموارساز گوسی

Uniform Additive Noise Filtered with Gaussian filter



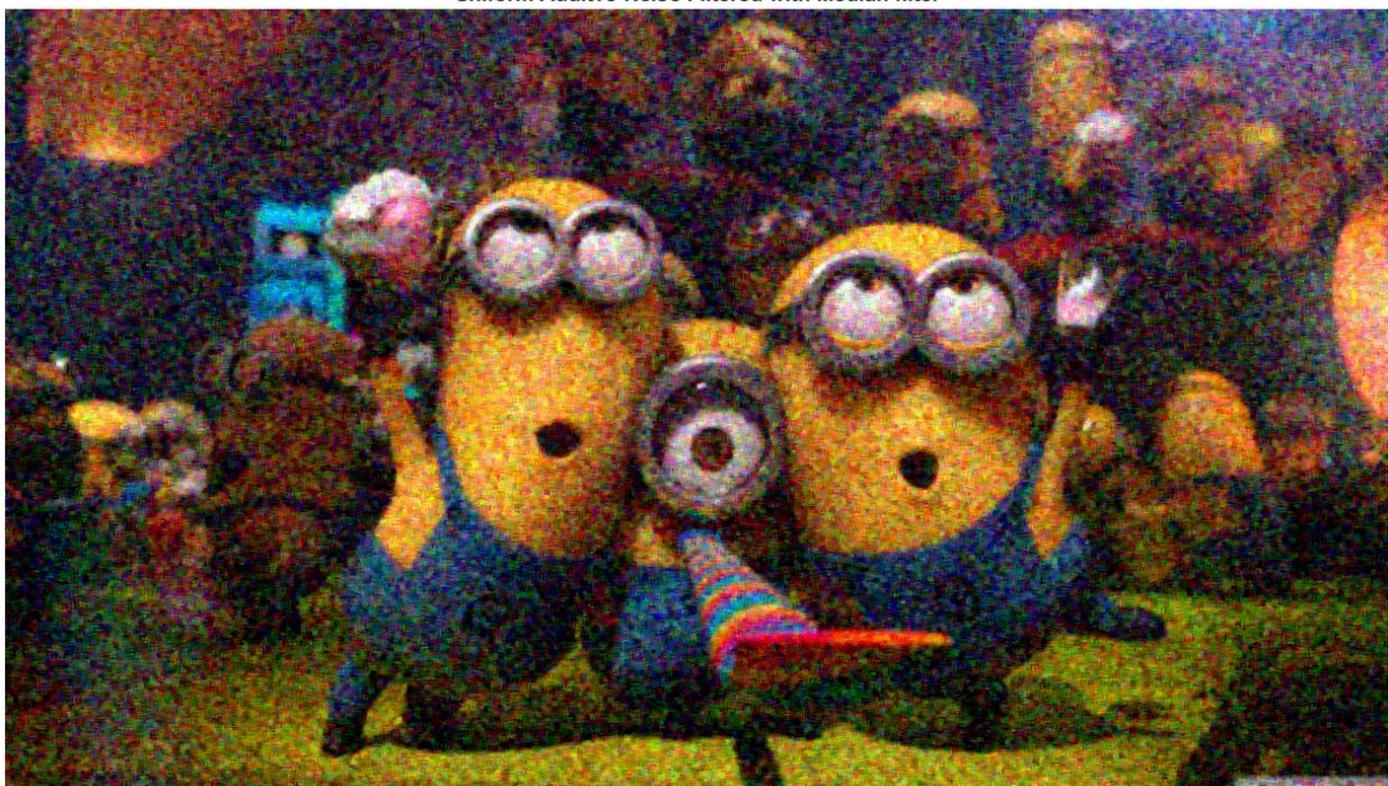
تصویر دوم، نویز جمع‌شونده یکنواخت، فیلتر wiener

Uniform Additive Noise Filtered with Wiener filter



تصویر دوم، نویز جمع‌شونده یکنواخت، فیلتر میانه

Uniform Additive Noise Filtered with Median filter



تصویر سوم، بدون نویز

Original Image



تصویر سوم، نویز جمع‌شونده گوسی

Additive White Gaussian Noise



تصویر سوم، نویز جمع‌شونده گوسی، فیلتر هموارساز میانگین متحرک

Additive White Gaussian Noise Filtered with moving average



تصویر سوم، نویز جمع‌شونده گوسی، فیلتر هموارساز گوسی

Additive White Gaussian Noise Filtered with Gaussian filter



تصویر سوم، نویز جمع‌شونده گوسی، فیلتر wiener

Additive White Gaussian Noise Filtered with Wiener filter



تصویر سوم، نویز جمع‌شونده گوسی، فیلتر میانه

Additive White Gaussian Noise Filtered with Median filter



Salt&Pepper Noise



تصویر سوم، نویز فلفل-نمکی، فیلتر هموارساز میانگین متحرک

Salt&Pepper Noise Filtered with moving average



تصویر سوم، نویز فلفل-نمکی، فیلتر هموارساز گوسی

Salt&Pepper Noise Filtered with Gaussian filter



تصویر سوم، نویز فلفل-نمکی، فیلتر wiener

Salt&Pepper Noise Filtered with Wiener filter



Salt&Pepper Noise Filtered with Median filter



تصویر سوم، نویز Shot

Shot (Poisson) Noise



### تصویر سوم، نویز Shot ، فیلتر هموارساز میانگین متحرک

Shot (Poisson) Noise Filtered with moving average



### تصویر سوم، نویز Shot ، فیلتر هموارساز گوسی

Shot (Poisson) Noise Filtered with Gaussian filter



Shot (Poisson) Noise Filtered with Wiener filter



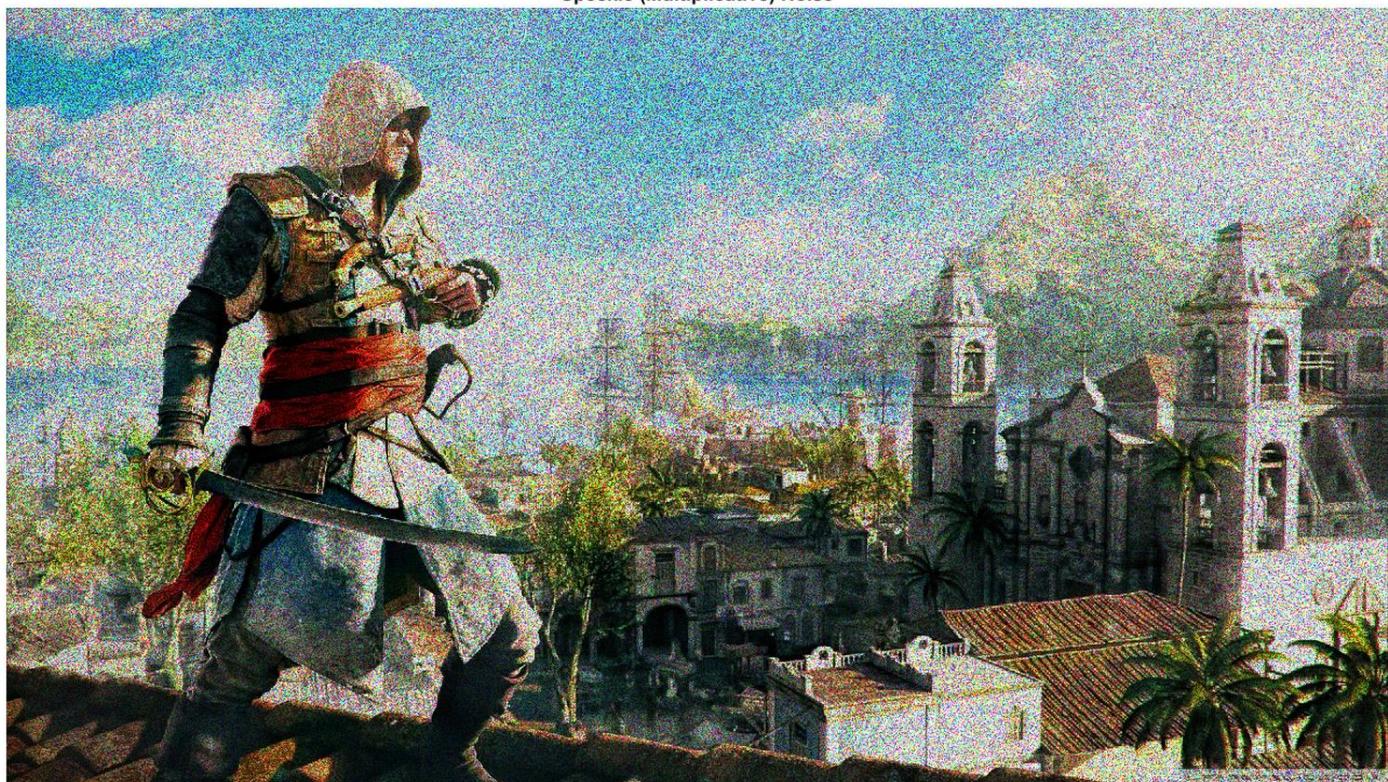
تصویر سوم، نویز Shot ، فیلتر میانه

Shot (Poisson) Noise Filtered with Median filter



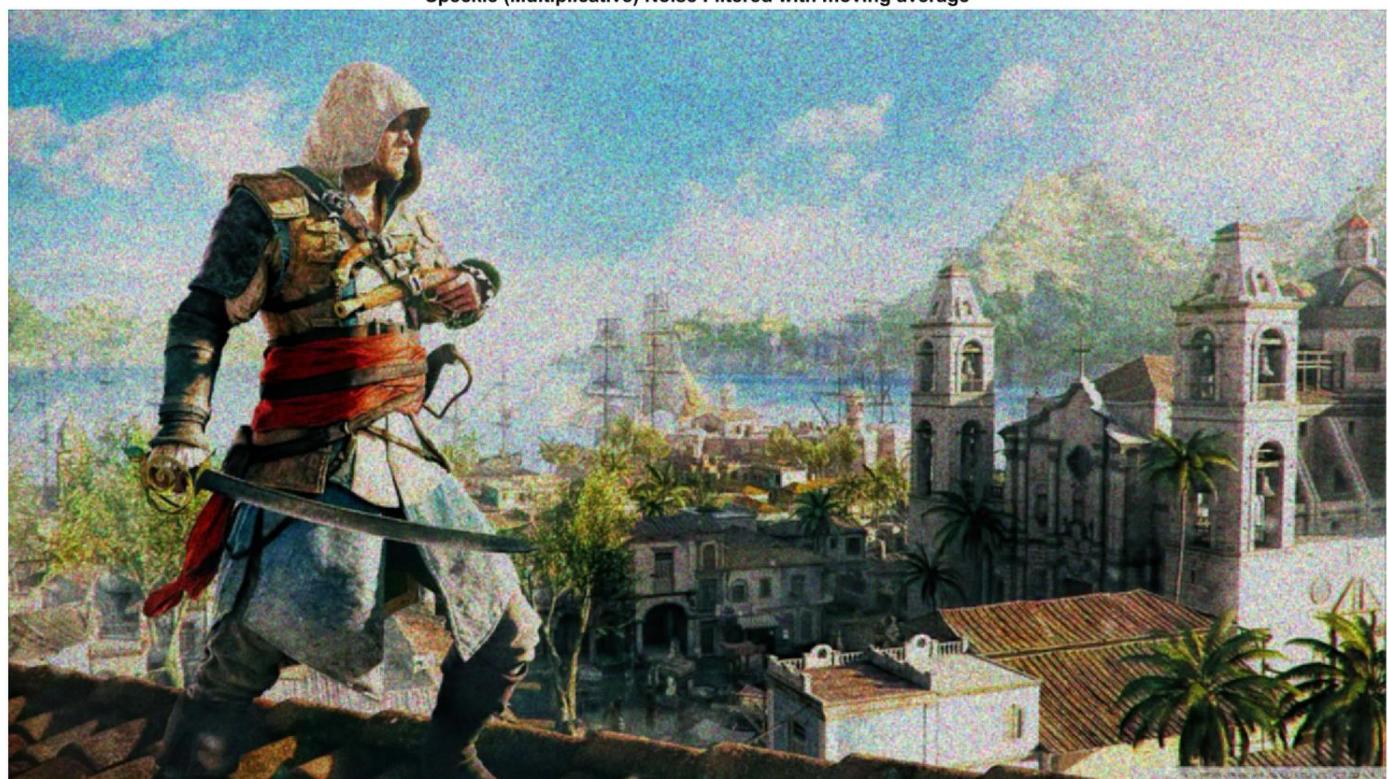
### تصویر سوم، نویز Speckle (ضرب‌شونده)

Speckle (Multiplicative) Noise



### تصویر سوم، نویز Speckle (ضرب‌شونده)، فیلتر هموارساز میانگین متحرک

Speckle (Multiplicative) Noise Filtered with moving average



### تصویر سوم، نویز Speckle (ضرب‌شونده)، فیلتر هموارساز گوسی

Speckle (Multiplicative) Noise Filtered with Gaussian filter



### تصویر سوم، نویز Speckle (ضرب‌شونده)، فیلتر wiener

Speckle (Multiplicative) Noise Filtered with Wiener filter



تصویر سوم، نویز Speckle (ضرب‌شونده)، فیلتر میانه

Speckle (Multiplicative) Noise Filtered with Median filter



تصویر سوم، نویز جمع‌شونده یکنواخت

Uniform Additive Noise



**تصویر سوم، نویز جمع‌شونده یکنواخت، فیلتر هموارساز میانگین متحرک**

Uniform Additive Noise Filtered with moving average



**تصویر سوم، نویز جمع‌شونده یکنواخت، فیلتر هموارساز گوسی**

Uniform Additive Noise Filtered with Gaussian filter



Uniform Additive Noise Filtered with Wiener filter



تصویر سوم، نویز جمع‌شونده یکنواخت، فیلتر میانه

Uniform Additive Noise Filtered with Median filter



**جمع‌بندی**

- فیلتر میانه بهترین عملکرد را برای نویز فلفل-نمکی دارد، چراکه در این نوع نویز، همه‌ی نقاط تصویر دچار نویز نمی‌شوند و با محاسبه‌ی میانه نقاط همسایه، با احتمال بسیار بالایی می‌توان مقداری بدون نویز را جایگزین مقدار نویزی کرد. این کیفیت بالای عملکرد، در تصاویر فوق نیز قابل مشاهده است.
- عملکرد wiener filter به جز در نویز فلفل-نمکی، در سایر موارد خوب بود. در مورد نویزهای shot و speckle، این فیلتر بهترین عملکرد را داشت، و در مورد نویزهای گوسی و یکنواخت، عملکردی مشابه با فیلتر میانگین متحرک داشت.
- دو فیلتر گوسی و میانگین متحرک، هر دو از خانواده فیلترهای هموارساز خطی انتخاب شدند، اما در اکثر موارد، عملکرد فیلتر میانگین متحرک، بهتر از فیلتر گوسی بود.
- نهایتاً برای تصمیم‌گیری، با توجه به عملکرد مشابه فیلتر wiener و فیلتر میانگین متحرک در برخی موارد، هر دو را در جدول زیر علامت می‌زنیم، اما دقت داریم که از خانواده فیلترهای هموارساز، ما دو نمونه را انتخاب کردیم، که یکی بعضًاً خوب و دیگری عملکرد نسبتاً پایین‌تری داشت، و ممکن است با انتخاب فیلترهای دیگری از مجموعه‌ی فیلترهای هموارساز خطی، نتایج بهتری بگیریم. نکته‌ی دیگری که در این مورد شایان ذکر است، آن است که طراحی فیلترهای بهینه برای رفع نویز در حال حاضر نیز مورد مطالعه است، و به عنوان نمونه می‌توان فیلتری هموارساز و خطی با هسته گوسی طراحی کرد که نویز سفید گوسی جمع‌شونده (AWGN) را به شکل بهینه برطرف کند. البته باید پارامترهای این فیلتر به شکل دقیق تعیین شود.
- مقایسه عملکرد این فیلترها زمینه‌ی پژوهش‌های بسیاری بوده است. به عنوان نمونه، ما مراجع زیر را نیز مورد مطالعه قرار دادیم:

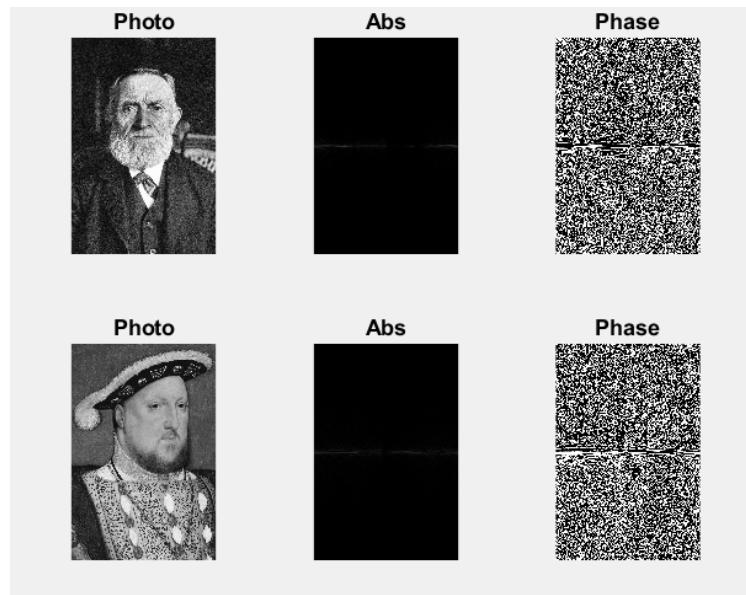
1. Survey of Image Denoising using Different Filters, Govindaraj.V, Sengottaiyan.G, International Journal of Science, Engineering and Technology Research (IJSETR) Volume 2, Issue 2, February 2013
2. Noise Types and Various Removal Techniques, Sukhjinder Kaur, International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Volume 4, Issue 2, February 2015
3. <https://ieeexplore.ieee.org/document/6133017/>

	Additive White Gaussian Noise	Salt&Pepper Noise	Shot Noise	Speckle Noise	Uniform Additive Noise
<b>Linear Smoothing Filter</b>	✓				✓
<b>Wiener Filter</b>	✓		✓	✓	✓
<b>Median Filter</b>		✓			

## بخش دوم - بررسی تاثیر اندازه و فاز در تبدیل فوریه

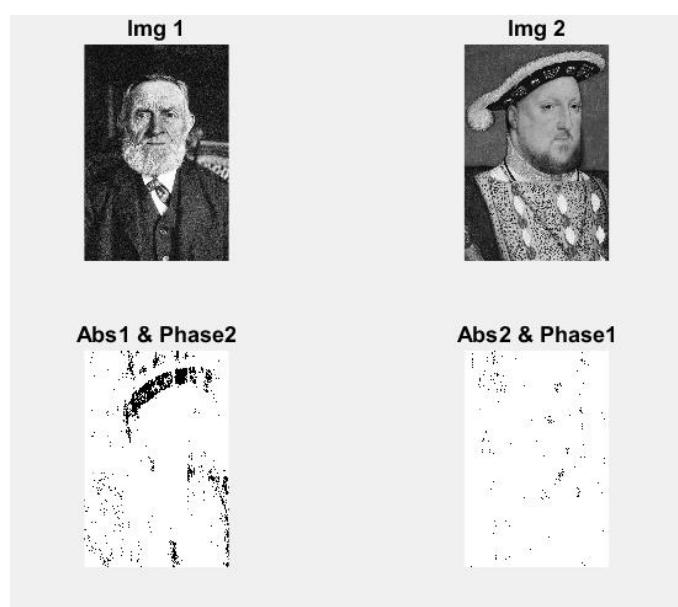
### • قسمت اول - پردازش تصاویر

ابتدا تبدیل فوریه سیگنال را به کمک روش FFT به دست می‌آوریم و اندازه و فاز آن را رسم می‌کنیم.



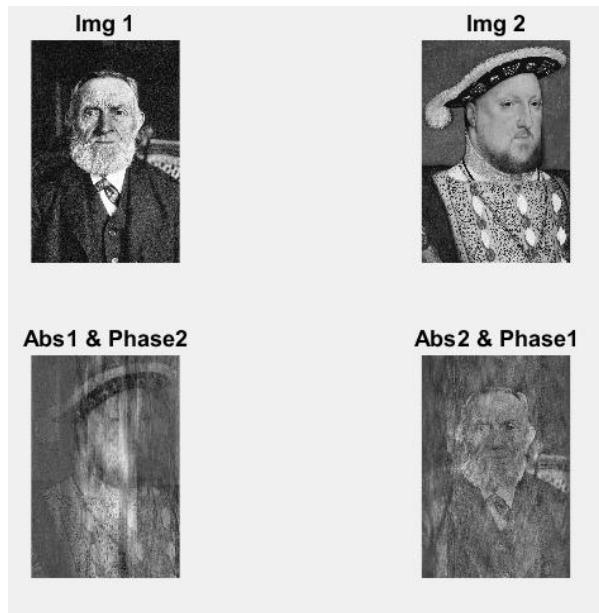
شکل ۲ - ۱ تبدیل فوریه تصاویر

حال تصاویر ۳ و ۴ را مطابق خواسته سوال تشکیل می‌دهیم.



شکل ۲-۲ تغییر ضربدری فازها و اندازه‌ها

به نظر می‌رسد که در تصاویر، فاز تاثیر بیشتری در درک ما از تصویر دارد. در مثال فوق، تصویری که مربوط به اندازه ۱ و فاز ۲ بود، شباهت زیادی به تصویر ۲ و تصویری که مربوط به اندازه ۲ و فاز ۱ بود، شباهت زیادی به تصویر ۱ داشت. برای اینکه این موضوع بیشتر نمایان شود، کانترسست تصویرها را زیاد می‌کنیم. نتایج زیر حاصل می‌شوند که موید ادعای ما هستند.



شکل ۲-۳ تغییر ضربدری فازها و اندازه‌ها با کانترس‌ت بالا

## • قسمت دوم—پردازش صوت

در ابتدا صورت singing.wav را لود کرده و تبدیل فوریه آن را محاسبه می‌کنیم. ابتدا فاز تبدیل فوریه این صورت را صفر کرده و تبدیل فوریه وارون را محاسبه می‌کنیم یعنی

$$\mathcal{F}^{-1}[|\mathcal{F}[x]|]$$

این کار بدین معناست که برای بازسازی توسط محتوای فرکانسی، از فاز سینوسی‌ها صرف نظر می‌کنیم. صوت حاصل از حذف فاز singing.wav در فایل NoPhase.wav آمده است. این صوت به طرز معناداری با صوت اولیه تفاوت دارد اما باز هم می‌توان درکی از صوت اولیه در آن پیدا کرد. به طور کلی ادعا این است که در صوت، بر خلاف تصویر، دامنه تاثیر بسیار بیشتری از فاز دارد. برای نمایش این موضوع از فاز و اندازه دو صوت به صورت ضربدری استفاده می‌کنیم.

این دو صوت فرکانس نمونه‌برداری‌های مختلف و همچنین طول‌های مختلفی دارند و برای اینکه بتوانیم عمل استفاده ضربدری از اندازه و فاز را انجام دهیم باید ابتدا فرکانس نمونه‌برداری و سپس طول دو صوت را یکسان کنیم.

فرکانس‌های نمونه‌برداری دو صوت ب.م.ای برابر با ۳۰۰ هرتز دارند که از فرکانس نایکوییست کمتر است پس نمی‌توانیم با یک سیستم down sampler معمولی فرکانس‌های نمونه‌برداری را برابر کنیم. پس ابتدا به کمک cubic spline دو صوت را درون یابی کرده و فرکانس نمونه‌برداری هر دو صوت را به کمک صوت آنالوگ حاصل به ک.م.دو فرکانس نمونه‌برداری اولیه یعنی ۷۰۵۶۰۰۰ هرتز می‌بریم و سپس به کمک یک سیستم down sampler آن را به مقدار ۷۸۴۰۰ کاهش می‌دهیم. اصوات حاصل از کیفیت بسیار مطلوبی برخوردار است. برای یکسان کردن طول دو صوت، بخش کوچکی از یکی از صوت‌ها را حذف می‌کنیم. ادامه عملیات را با این دو صوت که فرکانس نمونه‌برداری و طول یکسانی دارند انجام می‌دهیم.

گزارش کار تمرین متلب درس سیگنال‌ها و سیستم‌ها

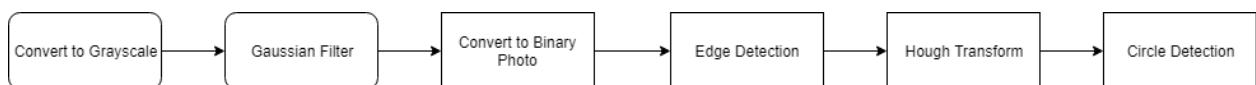
— سری پنجم —

صوت حاصل از تبدیل فوریه معکوس از اندازه sounds و فاز singing به اسم AbsSoundPhaseSing.wav و صوت حاصل از اندازه sounds و فاز singing در فایل AbsSingPhaseSound.wav ضمیمه شده است.

با مقایسه و بررسی این دو صوت به این نتیجه می‌رسیم که صوت حاصل به صوتی که اندازه تبدیل فوریه مشابه داشت شبیه‌تر است. این موضوع برخلاف نتیجه عمل مشابه در مورد تصویر است.

## بخش سوم – شمردن سکه‌ها در یک تصویر

در این بخش روشی برای تشخیص و شمردن سکه‌ها در یک تصویر ارائه می‌کنیم. روند کار در دیاگرام زیر ذکر شده است.



Moniri - Afsharrad

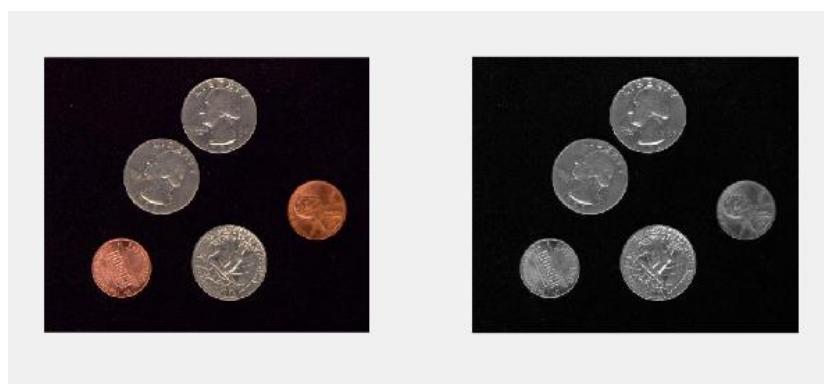
شکل ۱-۳ : دیاگرام بلوکی الگوریتم

حال به توضیح مراحل ششگانه بالا می‌پردازیم.

### ۱. تبدیل تصویر رنگی به سیاه و سفید

در این مرحله به کمک تبدیل زیر، عکس رنگی را به سیاه و سفید تبدیل می‌کنیم.

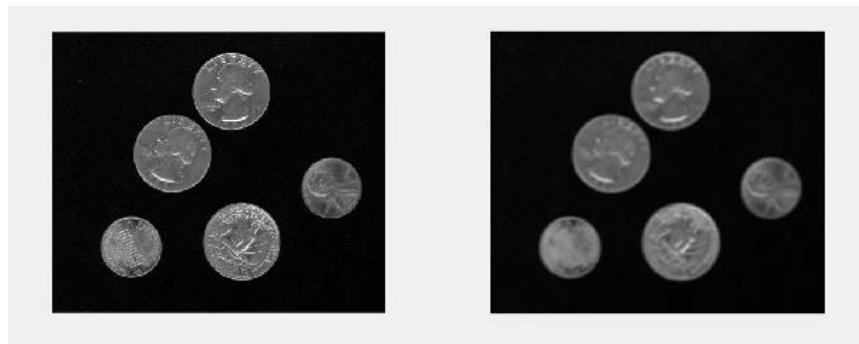
$$\text{Grayscale} = 0.2989 * \text{Red} + 0.5870 * \text{Green} + 0.1140 * \text{Blue}$$



شکل ۲-۳ : تبدیل تصویر رنگی به سیاه‌سفید

### ۲. اعمال فیلتر گاوی برای حذف نویز فرکانس بالای تصویر

در این بخش یک فیلتر گاوی را با میانگین صفر و واریانس ۴ بر تصویر اعمال می‌کنیم. این فیلتر مقدار تصویر در هر نقطه را با یک میانگین وزن دار از نقاط همسایه جایگزین می‌کند.



شکل ۳-۳ : اعمال فیلتر گاوی به تصویر سیاه‌سفید

	1	2	3	4	5
1	0.0232	0.0338	0.0383	0.0338	0.0232
2	0.0338	0.0492	0.0558	0.0492	0.0338
3	0.0383	0.0558	0.0632	0.0558	0.0383
4	0.0338	0.0492	0.0558	0.0492	0.0338
5	0.0232	0.0338	0.0383	0.0338	0.0232

شکل ۴-۳: ماتریس فیلتر گاوسی اعمال شده

### 3. تبدیل تصویر به یک تصویر باینری

در این بخش، نقاطی که شدت روشنایی آنها از یک آستانه بیشتر است را به روشنایی ۱ و دیگر نقاط را به ۰ نگاشت می‌کنیم. این آستانه به صورت تجربی به دست آمده است و ممکن است با توجه به نور محیط عکاسی و برخی دلایل دیگر، برای تصویرهای دیگر قابل استفاده نباشد و برای هر تصویر باید این آستانه به صورت مجزا به دست بیايد.



شکل ۴-۴: تبدیل تصویر به یک تصویر باینری

### 4. تشخیص لبه‌ها

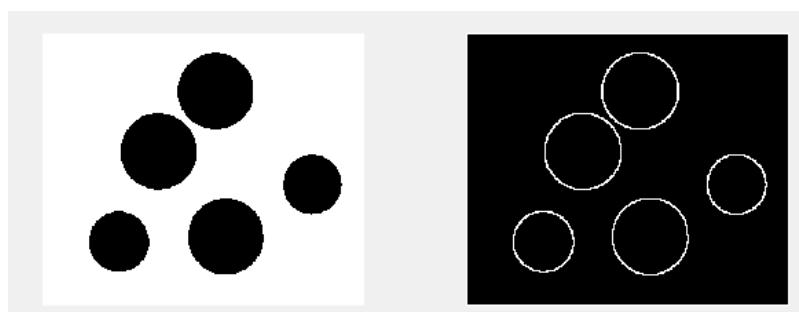
برای تشخیص لبه‌های تصویر باینری، یک تقریب عددی از گرادیان تصویر باینری ارائه می‌کنیم. نقاطی که این اندازه این گردیان ناصفر است، لبه‌های تصویر هستند.

برای تقریب  $\frac{\partial}{\partial x}$  از فیلتر  $L_x$  و برای تقریب  $\frac{\partial}{\partial y}$  از فیلتر  $L_y$  استفاده می‌کنیم.

$$L_x = [-1 \ 0 \ 1; -2 \ 0 \ 2; -1 \ 0 \ 1];$$

$$L_y = [1 \ 2 \ 1; 0 \ 0 \ 0; -1 \ -2 \ -1];$$

اندازه گرادیان برابر است با:  $\sqrt{(L_x * P)^2 + (L_y * P)^2}$ .



شکل ۴-۵: لبه‌های تصویر

## 5. تبدیل Hough

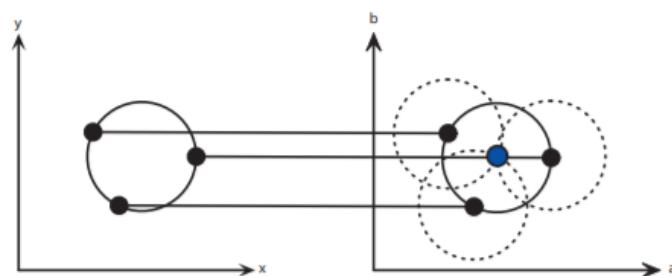
یک دایره به مرکز (a,b) و شعاع r در نظر بگیرید. معادله این دایره بدین صورت است.

$$\begin{cases} x = a + r \cos \theta \\ y = b + r \sin \theta \end{cases}$$

ابدا فرض کنید شعاع دایره برای ما معلوم و برابر r است. حال فرض کنید روی تمام (x,y) هایی که روی لبه دایره هستند، تبدیل

$$\begin{cases} a = x - r \cos \theta \\ b = y - r \sin \theta \end{cases}$$

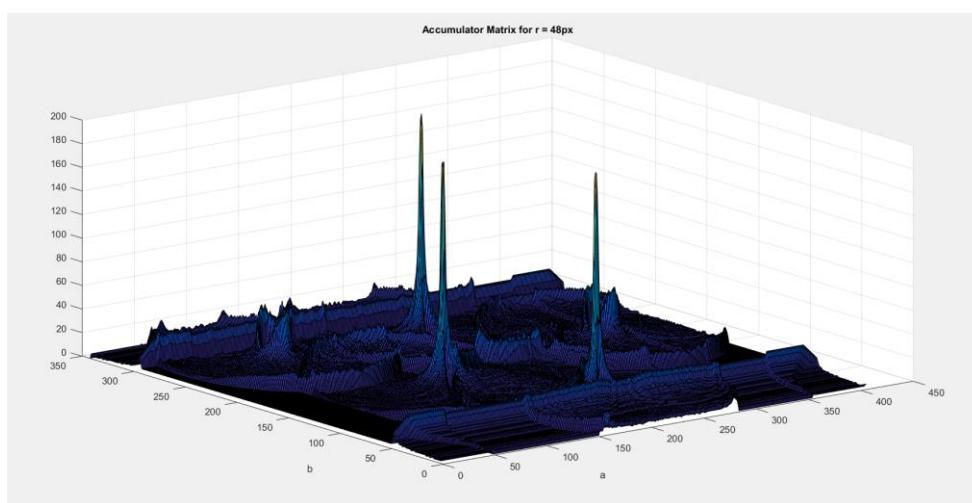
را اعمال کنیم بدین معنی که به ازای برای هر نقطه روی لبه شکل به یک (a,b) مشخص می‌رسم و مقدار ماتریسی به نام (a, b) یک واحد افزایش دهیم. به شکل زیر توجه کنید



شکل ۳-۷: یافتن مرکز دایره به کمک تبدیل هاف

این تبدیل نقاط را به دایره می‌برد. اگر این تبدیل را روی تمام نقاط دایره اصلی اعمال کنیم، در صفحه (a,b) به تعدادی دایره می‌رسیم که همگی در یک نقطه، که از قضا همان مرکز دایره اصلی است می‌رسیم پس انتظار داریم که اگر این تبدیل را، با یک شعاع خاص، بر کل تصویر اعمال کنیم و ماتریس Accumulator را رسم کنیم، این ماتریس در مراکز دایره‌ها به شعاع r بیشترین مقدار را داشته باشد. از این ایده و تکرار آن برای شعاع‌های مختلف برای پیدا کردن دایره‌ها در تصویر استفاده می‌کنیم.

ماتریس Accumulator برای شعاع ۴۸ پیکسل، به عنوان نمونه در زیر آمده است.



شکل ۳-۸ - ماتریس Accumulator

حال باید روشی ارائه دهیم که به کمک آن بتوانیم قله‌های این ماتریس را بشماریم. یک روش این است که آستانه‌ای پیدا کنیم که در صورتی که مقدار Accumulator در یک نقطه از آن آستانه بزرگتر شد، آن نقطه را به عنوان سکه معرفی کنیم. این روش یک ایراد اساسی دارد و آن این است که ممکن است یک سکه چند بار با این روش detect شود پس باید جواب به دست آمده را چک کنیم تا مطمئن شویم هر سکه را یک بار شمرده‌ایم و دایره‌هایی که مرکزشان از حدی به هم نزدیک‌تر است را حذف کنیم.

با اعمال این روش دایره‌هایی که باقی ماندند را رسم کرده و تعداد آنها را گزارش می‌کنیم.

Circles =

x0	y0	R
—	—	—
98	262	38
147	149	48
219	74	47
232	256	47
340	191	37

