گزارش پروژه پردازش تصاویر پزشکی

موضوع: تخمین و حذف نویز از تصاویر ام آر ای با استفاده از PCA غیرمحلی[1]

نویسنده: رضا کریم‌زاده

شماره دانشجویی: 98206234

استاد درس: دکتر فاطمی‌زاده

تابستان 99

[1] MRI noise estimation and denoising using non local PCA

فهرست

فهرست اشکال

1 مقدمه و معرفی مقاله

1_1 کلیت مطالعه

در این مطالعه از یک روش نوین برای حذف نویز از تصـــاویر MRI ارایه گردیده اســـت که بر اســـاس sparsity و تشابه میان تصاویر MRI عمل می‌کند. کلیت کار از دو مرحله تشکیل گردیده است. اول با تخمین نویز محلی و آســـتانه گذاری در PCA غیرمحلی تصـــویر حذف نویز می‌شـــود. در مرحله‌ی دوم تصویر فیلتر شده‌ی مرحله‌ی قبل به یک فیلتر تغییرناپذیر با دوران میانگین‌گیر غیرمحلی داده می‌شود تا تصویر حذف نویز شده‌ی نهایی بدست آید.

بنابراین با توجه به این‌که متد ارایه شده نویز را به صورت محلی تخمین می‌زند می‌توان آنرا به تصاویری که میزان نویز آنها در جاهای مختلف تصویر متفاوت است، اعمال نمود[1].

1_2 متدها

در این بخش به معرفی متدهای مورد استفاده در این پروژه پرداخته می‌شود.

1_2_1 حذف نویز با استفاده از PCA غیر محلی

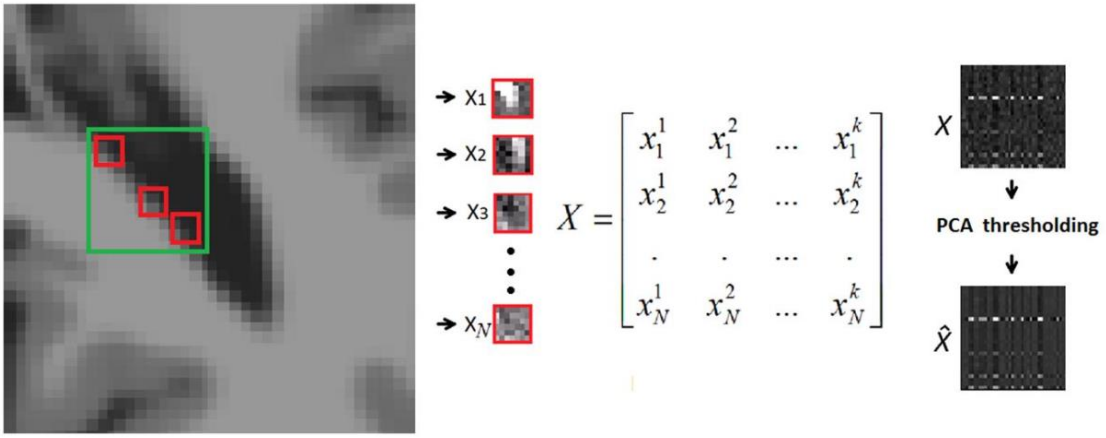
برای حذف نویز با استفاده از رویکرد PCA ابتدا لازم است پچ‌هایی که بیشترین شباهت را دارند در یک گروه قرار گیرند. این کار باعث تنکی بیشتر در نمایش این دسته پچ مشابه و در نتیجه تخمین بهتر نویز می‌شـــود. در این مطالعه پچ‌ها به صـــورت محلی در نظر گرفته شـــده و در نتیجه امکان تخمین نویز به صورت محلی ایجاد می‌شود. تعریف معمول تصویر نویزی به صورت یک تصویر تمیز که با یک نویز جمع شده است نشان داده می‌شود.

$$Y = A + N$$

که در آن Y تصویر نویزی، A تصویر بدون نویز و N نویز جمع شونده است. اکثر روش‌های حذف نویز به دنبال پیدا کردن A با دانستن Y هستند.

در متد ارایه شده در این مطالعه یک پنجره‌ی لغزان سه بعدی بر روی تصویر ام آر ای سه بعدی لغزانده می‌شود و با گروه کردن پچ‌های مشابه اطراف پچ اصلی یک ماتریس ساخته می‌شود. برای محدوده‌ی جستجو $(2t+1)^3$ پیکسل اطراف پیکسل مورد نظر استفاده می‌شود به این ترتیب با برداری کردن پچ‌های مشابه با پچ اصلی یک ماتریس N*K بوجود می‌آید که در آن N تعداد پچ‌های مشابه با پچ اصلی و K تعداد واکسل‌های هر پچ است. در این مطالعه N=K در نظر گرفته می‌شود بنابراین یک ماتریس N*N برای تبدیل PCA در دست است.

برای هر گروه از پچ‌ها این ماتریس ساخته می‌شود و بر روی آن PCA زده می‌شود. سپس المان‌های کم اهمیت، در واقع بردار ویژه‌های کوچک، با یک آستانه گذاری به این صورت که هر المان اگر از آستانه‌ی T کوچکتر بود حذف می‌شود. در ادامه تبدیل وارون PCA بر روی ماتریس آستانه‌گذاری شده محاسبه می‌شود و در نهایت چون پچ‌های انتخاب شده در پنجره‌های متفاوت وجود دارند باید یک ترکیب از نتایج مختلف در محل اصلی پچ جایگزین شود. برای این کار از میانگین گیری بین پچ‌ها استفاده می‌شود. شکل زیر این عملیات را نمایش می‌دهد.



شکل 1-1پچ بندی و آستانه گذاری بر روی PCA پچ‌های مشابه

نکته‌ی قابل توجه در انتخاب گروه پچ‌ها، انتخاب پچ‌های مشابه است. هرچه این پچ‌ها مشابه‌تر باشند نتیجه‌ی حذف نویز به خاطر تنکی در PCA پچ‌های مشابه بهتر خواهد بود. بنابراین استفاده از یک فیلتر پیش پردازش برای حذف نویز اولیه و انتخاب پچ‌های مشابه پیشنهاد می‌شود. فیلتر پیش پردازشی استفاده شده در این مطالعه، فیلتر میانه‌گیر سه بعدی است، به خصوص در مواقعی که نویز شدیدی وجود داشته باشد عملیات پچ‌بندی با استفاده از خروجی این فیلتر نتایج بهتری می‌دهد.

بنابراین بدون دانستن میزان نویز اولین پردازش میلنه‌گیری از تصویر است برای دادن به الگوریتم پچ‌بندی، پچ‌ها مشابه.

برای روشن‌تر شدن مفهوم و فرمول‌بندی PCA از یک مرجع دیگر [2] استفاده گردید. که در ادامه به توضیح نحوه‌ی عملکرد PCA پرداخته می‌شود. ماتریس x را به صورت زیر تعریف می‌کنیم.

$$x = [x_1\ x_2\ \dots x_m]^T$$

که در واقع هر درایه‌ی آن از یک بردار ساخته شده است.

$$\mathbf{X} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^n \\ x_2^1 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \vdots \\ x_m^1 & x_m^2 & \cdots & x_m^n \end{bmatrix}$$

ردیف iام ماتریس فوق را به صورت زیر تعریف می‌کنیم.

$$X_i = [x_i^1\ \ x_i^2\ \ \dots\ \ x_i^n]$$

حال برای هرکدام از این ردیف‌ها میانگین متناظر محاسبه می‌شود.

$$\mu_i = \frac{1}{n}\sum_{j=1}^{n} X_i(j)$$

در گام بعد با توجه به این میانگین‌ها سطر ماتریس x به مرکز منتقل می‌شود.

$$\overline{X}_i = X_i - \mu_i = [\overline{x}_i^1\ \ \overline{x}_i^2\ \ \dots\ \ \overline{x}_i^n]$$

این کار برای تمام ماتریس انجام می‌شود و در نهایت یک ماتریس مرکزی شده خواهیم داشت.

$$\overline{\mathbf{X}} = [\overline{X}_1^T\ \ \overline{X}_2^T\ \ \dots\ \ \overline{X}_m^T]^T$$

در گام بعد کوواریانس این ماتریس مرکزی شده به صورت زیر محاسبه می‌شود.

$$\mathbf{\Omega} = \frac{1}{n}\overline{\mathbf{X}}\overline{\mathbf{X}}^T$$

هدف الگوریتم PCA یافتن یک تبدیل عمودی P برای ناهمبسته سازی X است به بیان دیگر رابطه‌ی زیر بین P و X موجود است.

$$\overline{\mathbf{Y}} = \mathbf{P}\overline{\mathbf{X}}$$

بنابراین ماتریس کوواریانس P یک ماتریس قطری است. ماتریس کوواریانس Ω نیز متقارن است و می‌توان آن را به صورت زیر نوشت.

$$\mathbf{\Omega} = \mathbf{\Phi \Lambda \Phi^T}$$

که در آن ماتریس قطری مقادیر ویژه به صورت زیر است.

$$\mathbf{\Lambda} = diag\{\lambda_1, \lambda_2, \ldots, \lambda_m\}$$

بنابراین ماتریس P به صورت زیر بدست می‌آید.

$$\mathbf{P} = \mathbf{\Phi^T}$$

در نهایت با آستانه گذاری بر روی بردارهای ویژه و تبدیل وارون PCA می‌توان به داده‌ی بدون نویز رسید.

$$x' = PX + \mu$$

بنابراین در این بخش با عملکرد PCA و چگونگی حذف نویز از طریق آستانه گذاری بر روی بردارهای ویژه‌ی آن آشنا شدیم.

2_2_1 فیلتر میانگین‌گیر غیرمحلی تغییر ناپذیر با دوران

برای فیلتر میانگین‌گیر غیر محلی عادی میزان تشابه میان دو پچ i و j سنجیده می‌شد و بر اساس این میزان تشابه برای میانگین‌گیری بین آن‌ها یک میانگین وزن دار گرفته می‌شد.

$$\hat{x}(i) = \frac{\sum_{j \in \Omega} w(i,j)\, y(i)}{\sum_{j \in \Omega} w(i,j)} \qquad w(i,j) = e^{-\frac{\left\| N_i - N_j \right\|_2^2}{h^2}}$$

که در Ω حوزه‌ی تصویر که برای جستجوی پچ‌های مشابه در نظر گرفته شده، w معیار ارزیابی تشابه میان دو پچ و h یک پارامتر برای تنظیم قدرت فیلتر است.

همانطور که در فرمول بالا مشاهده می‌شود در صورتی که دو پچ کاملا مشابه باشند ولی دوران نسبت به یکدیگر داشته باشند وزن کمی در میانگین‌گیری می‌گیرند بنابراین برای حل این مشکل می‌توان دوران پچ‌ها را نسبت به یکدیگر بدست آورد و در نهایت با صفر کردن دوران میزان تشابه را سنجید. اما این روش یک مشکل اساسی دارد که بسیار این عملیات زمانگیر و هزینه‌ی محاسباتی بالایی دارد.

در این مطالعه یک روش بسیار ساده‌تر برای تغییر ناپذیر پچ‌ها نسبت به دوران ارایه گردیده است که در آن از شدت و میانگین واکسل‌ها استفاده می‌کند.

$$w(i,j) = e^{-\frac{1}{2}\left(\frac{(y(i)-y(j))^2+3(\mu_{N_i}-\mu_{Nj})^2}{2h^2}\right)}$$

که در آن μ_{N_i} و μ_{N_j} میانگین پچ‌های حول واکسل های i و j است. در رابطه‌ی بالا گذاشتن ضریب 3 برای میانگین پچ‌ها نشان دهنده‌ی دادن اهمیت بیشتر به میانگین پچ‌هاست و تجربه‌ی نویسنده‌ی مقاله نشان داده که عدد 3 بهترین عملکرد را دارد[3].

بنابراین این متد که ابتدا با *PCA* حذف نویز اولیه صورت گیرد و تصویر خروجی به فیلتر میانگین‌گیر غیر محلی تغییرناپذیر با دوران داده شود را *PRI-NL-PCA* نامگذاری می‌کنیم.

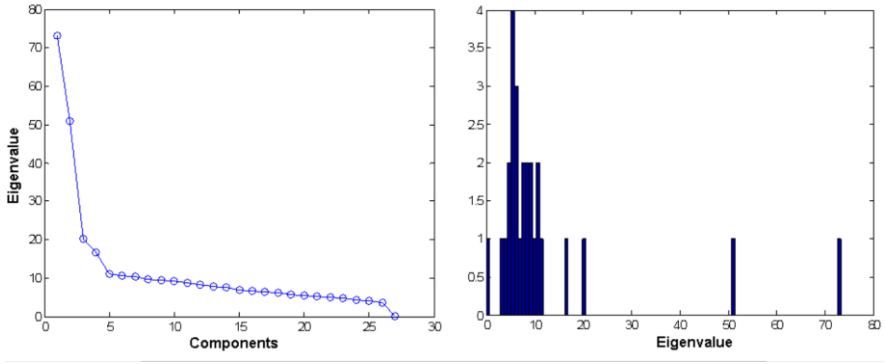
3_2_1 تخمین نویز بر اساس PCA

در حذف نویر با استفاده از الگوریتم PCA غیر محلی مهمترین نکته پیدا کردن آستانه‌ی مناسب برای حذف نویز است. این آستانه با انحراف معیار نویز در آن ناحیه‌ی نسبت مستقیم دارد. بنابراین تخمین نویز اساسی ترین قسمت این الگوریتم است و باید بسیار دقیق تعیین شود.

با توجه به این‌که در سیستم ام آر آی تصویر برداری به نحوی انجام می‌شود که در سراسر تصویر یک نویز همگن نداریم بنابراین استفاده از روش‌های تخمین نویز به صورت سراسری اشتباه است. در نتیجه باید نویز را به صورت محلی تخمین زد و آستانه‌گذاری بر اساس این تخمین در نواحی مختلف صورت گیرد.

در این مطالعه یک متد که کاملا با PCA غیر محلی تصـویر در ارتباط اسـت برای تخمین نویز اسـتفاده شـده اسـت. مقادیر ویژه‌ی تجزیه‌ی PCA با تغییرات سـیگنال و نویز در ارتباط اسـت به این صـورت که مقادیر ویژه‌ی بزرگتر متناسـب با سـیگنال هسـتند و مقادیر ویژه‌ی کوچکتر متناسـب با نویز هسـتند. بنابراین در اینجا ایده آسـتانه‌گذاری برای حذف نویز مطرح می‌شـود که این آسـتانه متناسـب با انحراف معیار نویز است.

همانطور که پیشتر اشاره گردید، هرچه پچ‌های انتخاب شده در یک ماتریس تشابه بیشتری داشته باشند تنکی در تبدیل PCA بیشتر است و سه عمده‌ی مقادیر ویژه متناظر با سیگنال است. حال برای تخمین میزان نویز می‌توان از یک میانه‌گیری بر روی مقادیر ویژه اسـتفاده نمود. به عنوان مثال در یک مجموعه پچ انحراف معیار نویز تقریبا 7 بوده اسـت. برای این مجموعه مقادیر ویژه محاسـبه گردیده اسـت و به صـورت زیر نشـان داده شـده اسـت. ملاحظه می‌شـود که با یک میانه‌گیری از مقادیر ویژه می‌توان با دقت قابل قبولی نویز را تخمین زد.



شکل 2-1نمودار مقادیر ویژه

بنابراین انحراف معیار نویز را می‌توان متناسب با میانه‌ی مقادیر ویژه دانست و می‌توان برای محاسبه‌ی آن از فرمول زیر بهره جست.

$$\hat{\sigma} = \beta\sqrt{median(\lambda)}$$

که در آن λ مقادیر ویژه و β ضریب تصحیح است که با تعداد واکسل‌های هر پچ و تعداد پچ‌های هر مجموعه نسبت دارد که همانطور که در بخش‌های پیش گفته شد در این پروژه تعداد پچ‌ها را با تعداد واکسل‌های هر پچ برای تبدیل PCA برابر در نظر می‌گیریم (N=K) بنابراین β متناظر با این نسبت 1.16 در نظر گرفته می‌شود.

این متد برای نویزهای با انحراف معیار کوچک و مجموعه پچ‌های شامل لبه نویز را بیشتر از حد موجود تخمین می‌زند بنابراین برای یک تخمین دقیق‌تر میانه‌گیری را بین مقادیر ویژه‌ی کوچکتر از میانه‌ی کلی محدود می‌کنیم بنابراین فرمول بندی تخمین نویز به صورت زیر ارایه می‌شود.

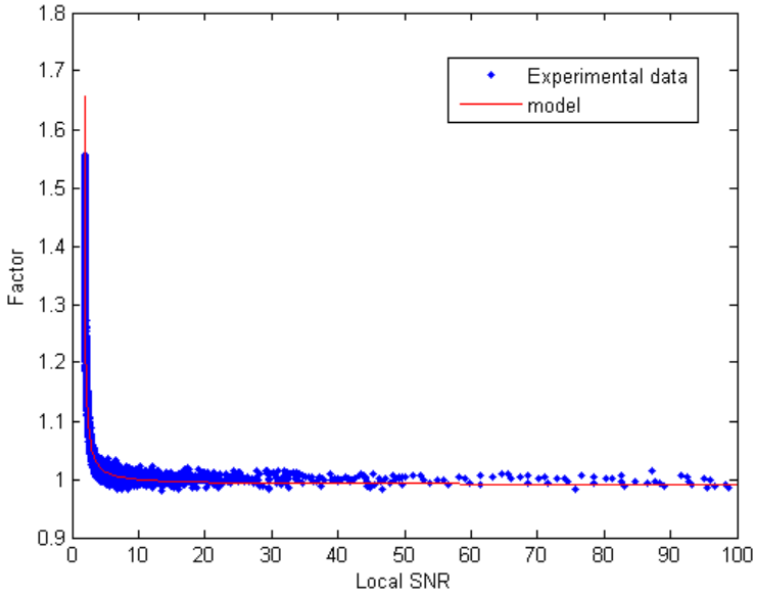
$$\hat{\sigma} = \beta\sqrt{median(\lambda_t)} \qquad \lambda_t = \{\lambda_i \mid \sqrt{\lambda_i} < 2median(\sqrt{\lambda})\}$$

4_2_1 تخمین نویز Rician

در اکثر فرض‌های حذف نویز، نویز را به صورت گوسی جمع شونده در نظر می‌گیرند. اما در تصاویر ام آر آی به صورت معمول نویز از توزیع Rician پیروی می‌کند. نامتقارنی توزیع Rician باعث ایجاد بایاس غیر ثابت در نواحی مختلف تصویر می‌شود بنابراین علاوه بر تخمین دقیق نویز در هر ناحیه باید بایاس مربوط به آن ناحیه نیز محاسبه شود.

در قسمت قبل انحراف معیار محاسبه شده برای نویز گوسی بود حال در این قسمت قصد داریم متدی ارایه دهیم که این انحراف معیار تخمین زده شده‌ی گوسی را به انحراف معیار نویز Rician تبدیل کنیم. این متد برخلاف سادگی آن بسیار عملکرد خوبی دارد. برای این تخمین از SNR محلی استفاده می‌شود.

برای محاسبه‌ی SNR محلی از تقسیم میانگین به انحراف معیار محلی استفاده می‌شود. در نهایت با استفاده از فرمول فیت شده بر آزمایشات SNR بر روی ضریب اصلاح انحراف معیار گوسی را به Rician تبدیل می‌کنیم.



شکل 3-1 آزمایشات انجام شده برای تخمین منحنی ضریب اصلاح

بنابراین این ضریب اصلاح بر حسب γ که همان SNR محلی تصویر است به صورت زیر محاسبه می‌شود.

$$\Phi(\gamma)=\begin{cases}\dfrac{((0.9846(\gamma-1.86)+0.1983)}{((\gamma-1.86)+0.1175))} & if\ (\gamma>1.86)\\ 0 \quad otherwise\end{cases}$$

در نهایت برای تخمین دقیق نویز Rician این ضریب تصحیح را در انحراف معیار تخمین زده شده ضرب می‌کنیم.

$$\hat{\sigma}=\sigma\,\Phi(\gamma)$$

5_2_1 تطابق با نویز Ricain

همانطور که پیشتر اشاره شد نویز تصاویر ام ار ای از نوع نویز Ricain است که در نواحی مختلف بایاس متفاوتی وجود دارد بنابراین لازم است این بایاس تصحیح شود. برای فرمول بندی این مفهوم، امید ریاضی چگالی احتمال نویز Ricain را می‌توان به صورت زیر نوشت.

$$E[R(v,\sigma)]=\sigma\sqrt{\frac{\pi}{2}}\exp\left(-\frac{v^2}{2\sigma^2}\right)\left(\left(1+\frac{v^2}{2\sigma^2}\right)I_0\left(\frac{v^2}{4\sigma^2}\right)+\left(\frac{v^2}{2\sigma^2}\right)I_1\left(\frac{v^2}{4\sigma^2}\right)\right)^2$$

که در آن v میانگین، σ انحراف معیار، I_0, I_1 توابع بسل نوع صفر و نوع اول هستند. با تغییر متغیر $\phi = v/\sigma$ فرمول زیر را خواهیم داشت.

$$\frac{\mathrm{E}[\mathrm{R}(\mathrm{v},\sigma)]}{\sigma} = \sqrt{\frac{\pi}{2}}\exp\left(-\frac{\phi^2}{2}\right)\left(\left(1+\frac{\phi^2}{2}\right)I_0\left(\frac{\phi^2}{4}\right)+\left(\frac{\phi^2}{2}\right)I_1\left(\frac{\phi^2}{4}\right)\right)^2$$

بنابراین می‌توان مقدار صحیح واکسل که بایاس آن تصحیح شده است را طبق رابطه‌ی زیر محاسبه نمود[4].

$$\hat{x} = \sigma\eta(x/\sigma)$$

در گام نهایی حذف بایاس نویز Ricain در فیلتر میانگین‌گیر غیرمحلی می‌توان از رابطه‌ی زیر استفاده نمود. که در آن σ انحراف معیار نویز Ricain تخمین زده شده در موقعیت i است.

$$\hat{A}(i) = \sqrt{\max\left(\left(\frac{\sum_{j\in\Omega} w(i,j)y(i)^2}{\sum_{j\in\Omega} w(i,j)}\right) - 2\sigma(i)^2, 0\right)}$$

3_1 خلاصه‌ی الگوریتم مورد استفاده

در این بخش خلاصه‌ای از متدهای مورد استفاده در این مطالعه و ترتیب پیاده‌سازی هر متد گفته می شود.

1_3_1 متد NL_PCA

1) پیاده‌سازی فیلتر میانه‌گیر سه بعدی بر روی تصویر نویزی
2) استفاده از تصویر خروجی مرحله 1 برای:
   - گروه‌بندی پچ‌های مشابه
   - تجزیه‌ی PCA بر روی این گروه ها اعمال می‌شود
   - با فرمول‌های گفته شده نویز تخمین زده می‌شود.
   - ترشلدگذاری برای حذف نویز

- محاسبه‌ی وارون PCA

3) ترکیب نتایج خروجی مرحله‌ی قبل با یکدیگر برای بدست آوردن تصویر حذف نویز شده و نویز تخمین زده شده در هر ناحیه
4) تصحیح بایاس نویز Rician

2_3_1 متد PRI_NL_PCA

1) اعمال متد NL_PCA و بدست آوردن تصویر خروجی
2) اعمال یک فیلتر میانگین‌گیر غیرمحلی و بدست آوردن نتیجه‌ی نهایی

4_1 دادگان مورد استفاده

دادگان مورد استفاده در این مطالعه، تصاویر ام آر آی T1_W[1] هستند که از فانتوم‌های BrianWeb استفاده شده است. به این تصاویر نویز گوسی و Rician اضافه گردید و سپس عملیات حذف نویز با متدهای بالا صورت داده شد.

[1] T1_wieghted

2 پیاده‌سازی متدهای مقاله

در این بخش به پیاده‌سازی الگوریتم‌ها و متدهای معرفی شده در بخش اول پرداخته می‌شود.

2_1 پیاده‌سازی فیلتر میانه‌گیر

برای این فیلتر ابتدا سایز کرنل مورد نظر و تصویر به ورودی تابع داده می‌شود و عملیات *padding* متناسب با ابعاد کرنل انجام می‌شود تا تصویر خروجی هم‌سایز تصویر ورودی باشد. سپس با لغزاندن کرنل در تصویر میانه‌ی کرنل به جای پیکسل مورد فیلتر قرار داده می‌شود و در نهایت تصویر فیلتر شده به خروجی می‌رود. این تابع توانایی میانه‌گیری سیگناهای یک، دو و سه بعدی را دارا می‌باشد. کد این فیلتر به صورت زیر است.

```
function B = medfilt3(A,siz,padopt,CHUNKFACTOR)
if nargin~=4
    CHUNKFACTOR = 1;
end
if CHUNKFACTOR<1, CHUNKFACTOR = 1; end

%% Checking input arguments
if isscalar(A), B = A; return, end

if ndims(A)>3
    error('A must be a 1-D, 2-D or 3-D array.')
end

if all(isnan(A(:))), B = A; return, end

sizA = size(A);
if nargin==1
    % default kernel size is 3 or 3x3 or 3x3x3
    if isvector(A)
        siz = 3;
    else
        siz = 3*ones(1,numel(sizA));
    end
    padopt = 'replicate';
elseif nargin==2
    % default padding option is "replicate"
    padopt = 'replicate';
end

%% Make SIZ a 3-element array
if numel(siz)==2
    siz = [siz 1];
elseif isscalar(siz)
    if sizA(1)==1
        siz = [1 siz 1];
    else
        siz = [siz 1 1];
    end
```

```
end

%% Chunks: the numerical process is split up in order to avoid large arrays
N = numel(A);
siz = ceil((siz-1)/2);
n = prod(siz*2+1);
if n==1, B = A; return, end
nchunk = (1:ceil(N/n/CHUNKFACTOR):N);
if nchunk(end)~=N, nchunk = [nchunk N]; end

%% Change to double if needed
class0 = class(A);
if ~isa(A,'float')
    A = double(A);
end

%% Padding along specified direction
% If PADARRAY exists (Image Processing Toolbox), this function is used.
% Otherwise the array is padded with scalars.
B = A;
sizB = sizA;
try
    A = padarray(A,siz,padopt);
catch
    if ~isscalar(padopt)
        padopt = 0;
        warning('MATLAB:medfilt3:InexistentPadarrayFunction',...
            ['PADARRAY function does not exist: '...
            'only scalar padding option is available.\n'...
            'If not specified, the scalar 0 is used as default.']);
    end
    A = ones(sizB+siz(1:ndims(B))*2)*padopt;
    A(siz(1)+1:end-siz(1),siz(2)+1:end-siz(2),siz(3)+1:end-siz(3)) = B;
end
sizA = size(A);

if numel(sizB)==2
    sizA = [sizA 1];
    sizB = [sizB 1];
end

%% Creating the index arrays (INT32)
inc = zeros([3 2*siz+1],'int32');
siz = int32(siz);
[inc(1,:,:,:) inc(2,:,:,:) inc(3,:,:,:)] = ndgrid(...
    [0:-1:-siz(1) 1:siz(1)],...
    [0:-1:-siz(2) 1:siz(2)],...
    [0:-1:-siz(3) 1:siz(3)]);
inc = reshape(inc,1,3,[]);

I = zeros([sizB 3],'int32');
sizB = int32(sizB);
[I(:,:,:,1) I(:,:,:,2) I(:,:,:,3)] = ndgrid(...
    (1:sizB(1))+siz(1),...
    (1:sizB(2))+siz(2),...
    (1:sizB(3))+siz(3));
I = reshape(I,[],3);

%% Check if NANMEDIAN exists
existNaNmedian = exist('nanmedian','file');

      %% Filtering
      for i = 1:length(nchunk)-1
```

```
102.
103.           Im = repmat(I(nchunk(i):nchunk(i+1),:),[1 1 n]);
104.           Im = bsxfun(@plus,Im,inc);
105.
106.           I0 = Im(:,1,:) +...
107.               (Im(:,2,:)-1)*sizA(1) +...
108.               (Im(:,3,:)-1)*sizA(1)*sizA(2);
109.           I0 = squeeze(I0);
110.
111.           if existNaNmedian
112.               B(nchunk(i):nchunk(i+1)) = nanmedian(A(I0),2);
113.           else
114.               B(nchunk(i):nchunk(i+1)) = median(A(I0),2);
115.           end
116.       end
117.       B = cast(B,class0);
```

2_2 پیاده‌سازی فیلتر مبتنی بر PCA

برای پیاده‌سازی این نوع فیلترینگ اولین تلاشی که صورت گرفت بر روی تصاویر دوبعدی خاکستری بود. برای این کار ابتدا به تصویر اصلی نویز گوسی با میانگین صفر اضافه شد سپس یک پنجره با ابعاد قابل تنظیم توسط کاربر و استفاده از دستور *im2col* متلب تمام پچ‌های تصویر در یک ماتریس قرار داده شدند، توجه شود این کار بدون توجه به پنجره‌های مشابه صورت گرفت و صرفا برای انجام فیلترینگ *PCA* انجام شد.

```
1.  clc;clear;close all
2.   s = 0.1             % noise standard deviation
3.   u = im2double(rgb2gray(imread('peppers.png')));
4.   v = u+s*randn(size(u));
5.
6.   f=5;
7.   Y = im2col(v,[2*f+1 2*f+1],'sliding');
```

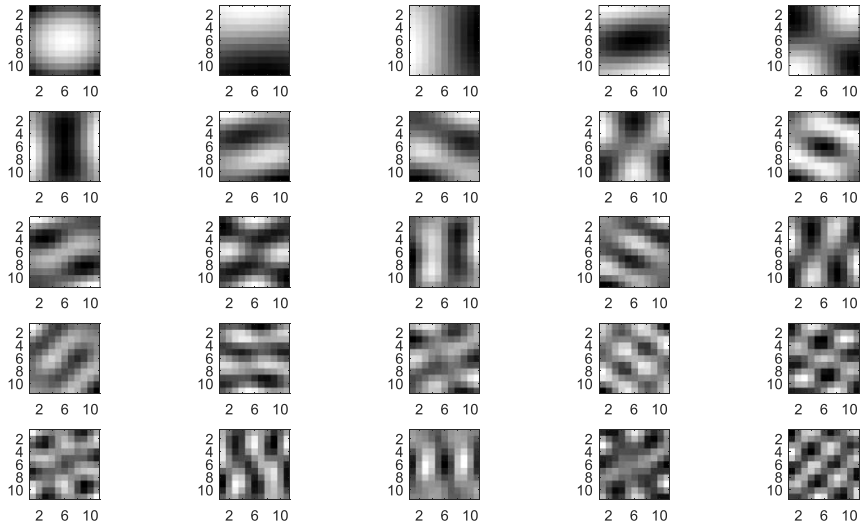
پس از استخراج ماتریس Y که شامل پچ‌های برداری شده تصویر است بر روی آن تبدیل *PCA* را همانطور که در بخش قبل توضیح داده شد، اعمال می‌کنیم. برای این کار ابتدا میانگین و کوواریانس ماتریس را بدست آورده و سپس مقادیر ویژه‌ی این ماتریس کوواریانس را محاسبه می‌کنیم. در گام بعد باید این مقادیر و بردارهای ویژه به ترتیب نزولی مرتب شوند.

```
1.   [nr, nc] = size(u);
2.   [mY,C,Yc]=moyCov(Y);
3.   [X,D] = eig(C);
4.   %%
5.   % plot eigenvalues in decreasing order
6.  [D,I] = sort(diag(D), 'descend'); plot(D);
7.  X = X(:,I);
8.  % plot first eigenvectors
9.  figure;colormap(gray);
```

```
for i=1:25
    subplot(5,5,i);imagesc(reshape(X(:,i),2*f+1,2*f+1));axis image;
end

function [mY,C,Yc]=moyCov(Y)
 mn = size(Y,2)
 mY  = (sum(Y,2)/mn);
 C = cov(Y');  %sum((Y-mY)*(Y-mY)')/mn
 Yc = minus(Y , mY);
end
```

برای داشتن یک دید کلی بردار ویژه‌های متناظر با مقادیر ویژه‌ی مرتب شده نشان داده شد.



شکل 1-2 بردارهای ویژه‌ی متناظر با مقادیر ویژه

مشاهده می‌شود که هرچه مقادیر ویژه کوچکتر می‌شوند بردارهای ویژه تغییرات بیشتری دارند و به نظر نویزی تر هستند.

در گام بعدی همانطور که در بخش اول اشاره شد برای فیلترگذاری در تبدیل PCA بر روی مقادیر ویژه آستانه‌گذاری صورت می‌گیرد و مقادیری که از آستانه کوچکتر هستند را حذف می‌کنیم. در اینجا آستانه 2.1 برابر انحراف معیار نویز قرار داده شده است. در نهایت تبدیل معکوس PCA برای رسیدن به ماتریس پچ‌های حذف نویز شده اعمال گردیده است.

```
Yproj=X'*Yc;                          % computation of all scalar products X_i^t
(Y_k-m_Y)
eta = 2.1*s;
Yproj(abs(Yproj)<eta)=0;              % hard thresholding
Z = X*Yproj + mY*ones(1,size(Y,2));   % reconstruction of the patches after hard
 thresholding
```

در مرحله‌ی نهایی این ماتریس پچ‌ها باید بازسازی شود تا به تصویر بدون نویز دست یافت. برای این کار پس از تغییر شکل پچ‌ها از بردار به ماتریس‌های با اندازه‌ی اصلی، یک میانگین‌گیری بین پچ‌های همپوشان صورت می‌گیرد و در نهایت تصویر اصلی بازسازی خواهد شد.

```
tmp = zeros(nr,nc,(2*f+1)^2);
nb = zeros(nr,nc);
for x = 1:2*f+1
  for y = 1:2*f+1
    i = (2*f+1) *(y-1)+x;
    w = reshape(Z(i,:),nr-2*f,nc-
2*f);          % use matrix instead of reshape in scilab
    tmp(x:nr-2*f+x-1,y:nc-2*f+y-1,i) = w;
    nb(x:nr-2*f+x-1,y:nc-2*f+y-1) = nb(x:nr-2*f+x-1,y:nc-2*f+y-1) +1;
  end
end
vdenoised = sum(tmp,3)./nb;
%%
figure;imshow([u, v, vdenoised],[])
```

نتیجه به صورت زیر حاصل شد. تصویر سمت راست، تصویر اولیه‌ی بدون نویز، تصویر وسط، تصویر نویزی با انحراف معیار 0.1 و تصویر سمت چپ، تصویر حذف نویز شده با استفاده از PCA است. مشاهده می‌شود نویز تا حد خوبی کاهش یافته است.

شکل 2-2 تصویر حذف نویز شده

حال همین روند برای تصاویر سه بعدی ام آر آی اعمال می‌شود. برای این کار همانطور که در PCA دو بعدی دیدیم لازم است از یک بلوک پچ‌های مشابه استخراج شوند. متلب برای استخراج پچ‌های دو بعدی دستور im2col را دارد اما برای پچ‌های سه بعدی باید این تابع نوشته شود بنابراین تابع im2col3D برای استخراج پچ‌های یک بلوک با سایز دلخواه و گام دلخواه نوشته شد.

```
function Y = im2col3D(im,r,step)
% extract patches with desired size and step
[m,n,w] = size(im);
counter = 1;
for i = 1:step:m-r+1
    for j = 1:step:n-r+1
```

```
        for q = 1:step:w-r+1
            temp = im(i:i+r-1,j:j+r-1,q:q+r-1);
            Y(:,counter) = temp(:);
            counter = counter +1;
        end
    end
end
end
```

در گام بعد پچ‌های برداری شده باید میانگین، پچ‌های به مرکز منتقل شده و کوواریانسشان محاسبه شود برای این کار از تابع زیر استفاده می‌شود.

```
function [mY, C ,Yc]=moyCov(Y)
 % function for calculating cov and mean vector
 mn = size(Y,2);
 mY  = (sum(Y,2)/mn);
 C = cov(Y');  %sum((Y-mY)*(Y-mY)')/mn
 Yc = minus(Y , mY);
end
```

پس از محاسبه‌ی ماتریس کوواریانس مقادیر ویژه‌ی آن محاسبه و مرتب می‌شود.

```
%% pca
Y = im2col3D(noisy_block,r,step);
[mY,C,Yc]=moyCov(Y);
 [X,D] = eig(C);
 %% eigenvalues in decreasing order
[D,I] = sort(diag(D), 'descend'); %plot(D);
X = X(:,I);
```

در گام بعد همانطور که در بخش اول گفته شد، باید تخمینی از نویز با فرمول‌های گفته شده بدست آوریم و با استفاده از این تخمین یک ترشلد برای حذف نویز با استفاده از PCA پیدا کنیم. برای این کار از تابع زیر استفاده می‌شود. ورودی این تابع مقادیر ویژه‌ی ماتریس کوواریانس است و با مراجعه به بخش اول برای محاسبه‌ی انحراف معیار نویز یک ضریب تصحیح (1.29) را در میانه‌ی مقادیر ویژه ضرب می‌کردند. این عملیات برای نویز گوسی صادق بود. در حالی که اگر نویز Rician بود باید یک عملیات دیگر با توجه به SNR بلوک صورت می‌گرفت تا به مقدار دقیق انحراف معیار محلی نویز دست یافته شود.

برای تمایز بین نویز گوسی و نویز Rician از یک بیت کنترلی در ورودی تابع استفاده شد تا در صورتی که نویز Rician بود عملیات تصحیح را برای این نوع نویز انجام دهد. کد تابع به صورت زیر است. خروجی این تابع انحراف معیار محلی نویز برای بلوک مورد نظر است.

```
function est_sigma = Noise_std_est(D,Rician,Y)
med = median(sqrt(D));
lambda_t = 0;
for i=1:length(D)
```

```
    if sqrt(D(i)) < 2*sqrt(med)
        lambda_t = [lambda_t D(i)];
    end
end
lambda_t = lambda_t(2:end);
beta = 1.29;
est_sigma = beta*sqrt(abs(median(lambda_t)));

if Rician
    l_m  = mean(Y();
    l_std = std(Y();
    gamma = l_m/l_std;
    if gamma >1.86
        phi_gamma  = (0.9846*(gamma-1.86)+0.1983)/((gamma-1.86)+0.1175);
    else
        phi_gamma  = 0;
    end
    est_sigma = est_sigma*phi_gamma;
end
end
```

در گام بعد با پیدا کردن میزان نویز محلی، عملیات آستانه گذاری انجام می‌شود. در تابع زیر مقادیر ویژه‌ی کوچکتر از 2.1 برابر نویز حذف می‌شوند و ماتریس پچ‌های حذف نویز شده بازسازی خواهند شد.

```
function Z = hrdThresh_vPatchRec(X,Yc,mY,Y,est_sigma)
Yproj=X'*Yc;                          % computation of all scalar products X_i^t
(Y_k-m_Y)
eta = 2.1*est_sigma;
Yproj(abs(Yproj)<eta)=0;              % hard thresholding
Z = X*Yproj + mY*ones(1,size(Y,2));   % reconstruction of the patches after hard
 thresholding
end
```

در آخر پس از حذف نویز پچ‌های برداری شده باید به مکان اصلی خود در بلوک مورد نظر برگردند. برای این کار از تابع زیر برای میانگین‌گیری پچ‌های همپوشان و ساخت ماتریس نهایی بلوک دینویز شده استفاده شد.

```
function vdenoised = block_reconstruction(Z,block_shape,r,step)
 nr = block_shape(1);nc = block_shape(2);nw = block_shape(3);
 tmp = zeros(nr,nc,nw,(nr*nc*nw));
nb = zeros(nr,nc,nw);
i=0;
for x = 1:step:nr-r+1
  for y = 1:step:nc-r+1
      for l =1:step:nw-r+1
    i = i+1;
    w = reshape(Z(:,i),r,r,r);           % use matrix instead of reshape in scila
b
    tmp(x:r+x-1,y:r+y-1,l:r+l-1,i) = w;
    nb(x:r+x-1,y:r+y-1,l:r+l-1) = nb(x:r+x-1,y:r+y-1,l:r+l-1) +1;
  end
  end
end
vdenoised = (sum(tmp,4))./nb;
 end
```

بنابراین تا این مرحله توانستیم برای یک بلوک پچ‌های مشابه را جدا کنیم و عملیات حذف نویز و تخمین نویز را برای بلوک مورد نظر انجام دهیم. در مقاله، ابعاد هر بلوک 7*7*7 و ابعاد هر پچ 4*4*4 در نظر گرفته شده است که پچ‌ها می‌توانند با گام‌های مختلف انتخاب شوند. بنابراین با استفاده از توابع معرفی شده در بالا برای نظم بیشتر کد، یک تابع نوشته می‌شود که عملیات حذف نویز برای هر بلوک را انجام می‌دهد و در خروجی بلوک حذف نویز شده و انحراف معیار نویز تخمین‌زده شده را می دهد.

```
function  [denoised_block, block_map] = Nl_PCA_blockwise(noisy_block,r,step,Rician)
%% pca
Y = im2col3D(noisy_block,r,step);
[mY,C,Yc]=moyCov(Y);
 [X,D] = eig(C);
 %% eigenvalues in decreasing order
[D,I] = sort(diag(D), 'descend'); %plot(D);
X = X(:,I);
%% Noise estimation
est_sigma = Noise_std_est(D,Rician, Y);

%% hard thresholding and vector patch reconstruction
Z = hrdThresh_vPatchRec(X,Yc,mY,Y,est_sigma);

 %% reconstructing original block
block_shape = size(noisy_block);
denoised_block = block_reconstruction(Z,block_shape,r,step);
block_map = est_sigma*ones(block_shape);
end
```

طبق آنچه گفته شد بایست برای بلوک‌های 7*7*7 کل تصویر پچ‌های 4*4*4 در این بلوک‌ها حذف نویز شود و در نهایت تجمیع و میانگین‌گیری بین شدت و میزان نویز بلوک‌ها برای ساخت تصویر نویزی و نقشه‌ی نویز در تصویر صورت گیرد. با توجه به کد زیر ابتدا این عمل برای تمامی بلوک‌ها انجام می‌شود و عملیات تجمیع به صورت همزمان صورت می‌پزیرد. در مرحله‌ی بعد در صورت وجود نویز Rician بایاس آن باید حذف شود بنابراین از فرمول زیر برای حذف این بایاس استفاده شد.

$$\hat{x} = \sigma\eta(x/\sigma)$$

```
function [final_img, map] = NL_PCA(noisy_im,r,t,step,Rician)
s = size(noisy_im);
denoised_img = zeros(s);
normalization = zeros(s);
map = zeros(s);
for ii=1:s(1)-2*t
    for jj=1:s(2)-2*t
        for ll = 1:s(3)-2*t
            noisy_block = noisy_im(ii:ii+2*t,jj:jj+2*t,ll:ll+2*t);
            [denoised_block, block_map] = Nl_PCA_blockwise(noisy_block,r,step,Rician);
```

```
            denoised_img(ii:ii+2*t,jj:jj+2*t,ll:ll+2*t) =denoised_block +  denoised_img(ii:ii+2*t,jj:jj+2*t,ll:ll+2*t);
            map(ii:ii+2*t,jj:jj+2*t,ll:ll+2*t) =block_map +  map(ii:ii+2*t,jj:jj+2*t,ll:ll+2*t);
            normalization(ii:ii+2*t,jj:jj+2*t,ll:ll+2*t) =1 +  normalization(ii:ii+2*t,jj:jj+2*t,ll:ll+2*t);
        end
    end
end
final_img = denoised_img./normalization;
map = map./normalization;
if Rician
bias_corrected_img = zeros(s);
mdi = max(final_img(:));
for ii=1:s(1)
    for jj=1:s(2)
        for ll = 1:s(3)
           bias_corrected_img(ii,jj,ll)= map(ii,jj,ll) * etta( (final_img(ii,jj,ll)/(map(ii,jj,ll)*mdi)));
        end
    end
end
end
end

function out = etta(phi)
phi24 = (phi^2)/4;
out =( sqrt(pi/2)*exp(-phi24)*((1+2*phi24)*besseli(0,phi24) + (2*phi24)*besseli(1,phi24))^2);
end
```

بنابراین تا این مرحله موفق به پیاده سازی فیلتر NL_PCA شدیم که خروجی آن تصویر حذف نویز شده و یک نقشه از انحراف معیار نویز در نقاط مختلف تصویر است.

در گام آخر فیلتر میانگین‌گیر غیر محلی تغییر ناپذیر نسبت به دوران برای تصویر سه بعدی پیاده‌سازی می‌شود. در این فیلتر از فرمول زیر استفاده شد.

$$w(i,j) = e^{-\frac{1}{2}\left(\frac{(y(i)-y(j))^2+3(\mu_{N_i}-\mu_{Nj})^2}{2h^2}\right)}$$

که برای h باید مقداری متناسب با شدت نویز در پیکسل مورد نظر انتخاب شود به همین علت نقشه‌ی نویز بدست آمده در مرحله‌ی قبل باید به این فیلتر داده شود. در مقاله ذکر شده که h می‌تواند 0.4 انحراف معیار نویز باشد. در گام بعدی این فیلترگذاری با استفاده از فرمول زیر درصورتیکه نویز Ricain وجود داشته باشد اقدام به حذف بایاس نهایی می‌شود.

$$\hat{A}(i) = \sqrt{\max\left(\left(\frac{\sum_{j\in\Omega} w(i,j)y(i)^2}{\sum_{j\in\Omega} w(i,j)}\right) - 2\sigma(i)^2, 0\right)}$$

کد این قسمت به صورت زیر می‌باشد.

```
function denoised = PRI_NL_PCA(img,v,t,map,Rician)
B = padarray(img,[v, v, v],'symmetric');
s = size(B);
denoised = zeros(size(img));
for i = v+1:s(1)-v
    i-v
    for j = v+1:s(2)-v
        for l = v+1:s(3)-v
          search_blk = B(i-v:i+v, j-v:j+v, l-v:l+v);
          ref_patch = B(i-t:i+t, j-t:j+t, l-t:l+t);

          sig_wiegh = 0;
          sig_patch_wiegh = 0;
          h = 0.4*map(i-v,j-v,l-v);
          for k=t+1:2*v-t+1
              for q=t+1:2*v-t+1
                  for w=t+1:2*v-t+1
                      patch = search_blk(k-t:k+t,q-t:q+t,w-t:w+t);

                      gi = (ref_patch(t+1,t+1, t+1)-patch(t+1,t+1, t+1))^2;
                      ui =3*(mean(ref_patch(:))-mean(patch(:)))^2;


                      wiegh = exp(-0.5*((gi+ui)/2*h^2));
                      sig_wiegh = sig_wiegh + wiegh;
                      if Rician
                          patch_wiegh = wiegh*(patch(t+1,t+1,t+1))^2;
                      else
                      patch_wiegh = wiegh*patch(t+1,t+1,t+1);
                      end
                      sig_patch_wiegh = sig_patch_wiegh+patch_wiegh;
                  end
              end
          end
          if Rician
              denoised(i-v,j-v,l-v) =sqrt(max( (sig_patch_wiegh/sig_wiegh)-
2*(map(i-v,j-v,l-v))^2,0));
          else
          denoised(i-v,j-v,l-v) = sig_patch_wiegh/sig_wiegh;
          end
        end
    end
end

end
```

این کد به علت جستجو برای یافتن پچ‌های مشابه هزینه‌ی محاسباتی بالایی دارد.

3 نتایج و مقایسه با مقاله اصلی

برای مقایسه‌ی نتایج با مقاله اصلی، لازم است نویز گوسی و Rician در دوحالت ثابت و متغییر با مکان به تصویر اضافه شوند و علاوه بر اعتبار سنجی حذف نویز توسط فیلترهای معرفی شده، میزان دقت تخمین نویز نیز محاسبه شود. بر این اساس برای اعتبار سنجی حذف نویز از معیارهای PSNR و SSIM و برای اعتبار سنجی تخمین میزان نویز ER و MER استفاده می‌گردد که در ادامه به توضیح هر یک پرداخته می‌شود.

3_1 معیارهای اعتبار سنجی

3_1_1 پیک نسبت سیگنال به نویز (PSNR)

این نسبت نیز مانند نسبت سیگنال به نویز است با این تفاوت که مقدار ماکزیمم تصویر بدون نویز را به جای مجموع تمام شدت‌ها در نظر می‌گیرد. هرچقدر این معیار بزرگتر باشد عملکرد فیلتر بهتر است. در زیر فرمول آن مشاهده می‌شود.

$$PSNR(x,y) = 10log_{10}(\frac{L^2}{\frac{1}{MN}\Sigma_{i,j}\big(x(i,j)-y(i,j)\big)^2})$$

که در آن L بیشترین مقدار شدت در تصویر تمیز است و M, N ابعاد تصویر هستند

3_1_2 معیار SSIM[1]

این معیار تقریبا با معیارهای بینایی انسان برابر است. فرمول آن به صورت زیر است.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)},$$

[1] Structural Similarity Index

که در آن μ_x و μ_y میانگین تصویر تمیز و نویزی است. σ_x و σ_y انحراف معیار تصویر تمیز و نویزی و σ_{xy} کوواریانس دو تصویر است. C1, C2 به صورت زیر تعریف می‌شوند.

$$c_1 = (k_1L)^2, \text{ and } c_2 = (k_2L)^2$$

که در آن L رنج دینامیکی تصویر و $K1, K2$ مقادیر ثابتی هستند($K1 = 0.01, K2 = 0.03$).

3_1_3 اعتبار سنجی تخمین نویز

برای این اعتبار سنجی از نسبت خطا[1] استفاده می‌شود که برای یک پیکسل که در آن تخمین نویز انجام شده است به صورت زیر محاسبه می‌شود.

$$\mathrm{ER} = \left|1 - \hat{\sigma}/\sigma\right|$$

حال این نسبت برای تمام پیکسل‌ها محاسبه می‌شود و میانگین‌گیری انجام می‌شود.

$$\mathrm{MER} = \frac{1}{|\Omega|}\sum_{\forall i \in \Omega}\left|1 - \hat{\sigma}_i/\sigma_i\right|$$

3_2 اضافه کردن نویز و فیلترکردن تصویر نویزی

همانطور که پیشتر اشاره شد لازم است نویز گوسی و Rician در دوحالت ثابت و متغییر با مکان به تصویر اضافه شوند برای این کار ابتدا تصویر بدون نویز سه بعدی خوانده شد و سپس با توجه به سویچ بیت‌هایی نوع نویز و ثابت یا متغییر بودن آن تعیین می‌شود. در ادامه با استفاده از یک حلقه‌ی for درصدهای مختلف نویز به تصویر اضافه می‌شود و با استفاده از فیلترهای معرفی شده در بخش پیش حذف نویز و تخمین نویز صورت میگیرد در نهایت معیارهای گفته شده محاسبه و نمایش داده می‌شوند. کد زیر مربوط به این عملیات است.

```
% read volume
name ='t1_icbm_normal_1mm_pn0_rf0.rawb';
fid = fopen(name,'r');
s=[181,217,181];
```

[1] Error Ratio

```
ima=zeros(s(1:3));
for z=1:s(3),
  ima(:,:,z) = fread(fid,s(1:2),'uchar');
end;
fclose(fid);
ima=double(ima);


% sigma = 20;
Rician = 0 %rician distribution 1 & gaussian 0
variable = 0 %variable=0(Homogeneus noise)  variable=1(spatially variable noise)

% subvolume (do a test with a smaller volume)
ima=ima(50:60,50:60,50:60);
s=size(ima);
for i=1:2:9
  i
sigma=i*max(ima(:))/100;
randn('seed',0)
if(variable)
  map = ones(3,3,3);
  map(2,2,2)=3;
  [x1,y1,z1] = meshgrid(1:3,1:3,1:3);
  [x2,y2,z2] = meshgrid(1:2/(s(2)-1):3,1:2/(s(1)-1):3,1:2/(s(3)-1):3);
  map = sigma*interp3(x1,y1,z1,map,x2,y2,z2,'cubic');
  if(Rician) rima=sqrt((ima+randn(size(ima)).*map).^2+(randn(size(ima)).*map).^2)
;
  else       rima=ima+randn(size(ima)).*map;
  end
else
  if(Rician) rima=sqrt((ima+randn(size(ima))*sigma).^2+(randn(size(ima))*sigma).^
2);
  else       rima=ima+randn(size(ima))*sigma;
  end
  map=ones(s)*sigma;
end


%% Median Filter
med_filt = medfilt3(rima,[3 3 3]);
%% Non Local PCA

t = 3; %block half size (2*t+1)
r = 4;  % patch size
step = 1;
[denoised_step1, map_est] = NL_PCA(med_filt,r,t,step,Rician);
%% Non Local mean
v = 5 %half size of search window
t = 1  %half size of surrounding window in pixel (i,j,k)
denoised_step2 = PRI_NL_PCA(denoised_step1,v,t,map_est,Rician);
%% show result
figure;
clf
colormap(gray);
n=round(size(denoised_step1,3)/2);
suptitle(['Results for ', num2str(i), '% Noise Level'])
subplot(2,2,1),imagesc(imrotate(ima(:,:,n),90));title('Original Noise free Image'
)
subplot(2,2,2),imagesc(imrotate(rima(:,:,n),90));;title('Noisey Image')
subplot(2,2,3),imagesc(imrotate(denoised_step1(:,:,n),90));;title('NL-
PCA denoised')
```

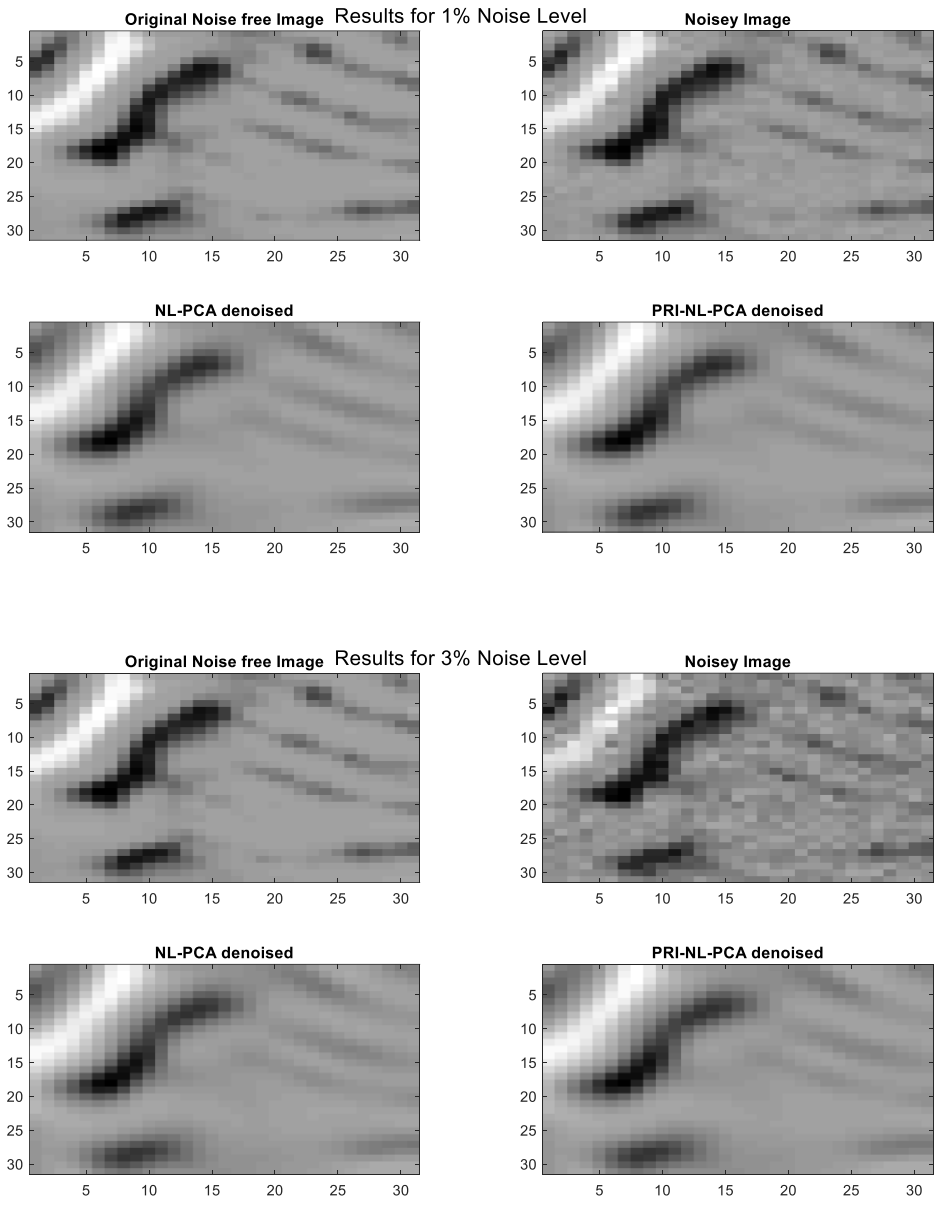
```
63. subplot(2,2,4),imagesc(imrotate(denoised_step2(:,:,n),90));;title('PRI-NL-
    PCA denoised')
64. %% fidelity
65. % find max intensity of image for psnr
66. R=max(ima(:));
67. indi=find(ima>10);
68. sw = [1 1 1];
69. % for noisy image
70. oerror0(i)=sqrt(mean((ima(indi)-rima(indi)).^2));
71. opsnr0(i)=20*log10(R/oerror0(i));
72. ossim0(i)= ssim_index3d(rima,ima,sw,(indi));
73. % for NL_PCA
74. oerror1(i)=sqrt(mean((ima(indi)-denoised_step1(indi)).^2));
75. opsnr1(i)=20*log10(R/oerror1(i));
76. ossim1(i)= ssim_index3d(denoised_step1,ima,sw,indi);
77. % for PRI-NL-PCA
78. oerror2(i)=sqrt(mean((ima(indi)-denoised_step2(indi)).^2));
79. opsnr2(i)=20*log10(R/oerror2(i));
80. ossim2(i)= ssim_index3d(denoised_step2,ima,sw,indi);
81.
82. ER = abs(1-map_est./map);
83. MER(i) = mean(ER(:));
84. end
85. %% plots
86. op0=mean(opsnr0(1:2:9))
87. op1=mean(opsnr1(1:2:9))
88. op2=mean(opsnr2(1:2:9))
89.
90. figure
91. title('PSNR for noisy, NL-PCA and PRI-NL-PCA output')
92. clf
93. plot(1:2:9,opsnr0(1:2:9),'g')
94. hold on
95. plot(1:2:9,opsnr1(1:2:9),'b')
96. plot(1:2:9,opsnr2(1:2:9),'r')
97. xlabel('Noise level(%)')
98. ylabel('PSNR')
99.
100.        os0=mean(ossim0(1:2:9))
101.        os1=mean(ossim1(1:2:9))
102.        os2=mean(ossim2(1:2:9))
103.
104.        figure
105.        title('SSIM for noisy, NL-PCA and PRI-NL-PCA output')
106.        clf
107.        plot(1:2:9,ossim0(1:2:9),'g')
108.        hold on
109.        plot(1:2:9,ossim1(1:2:9),'b')
110.        plot(1:2:9,ossim2(1:2:9),'r')
111.        xlabel('Noise level(%)')
112.        ylabel('SSIM')
```

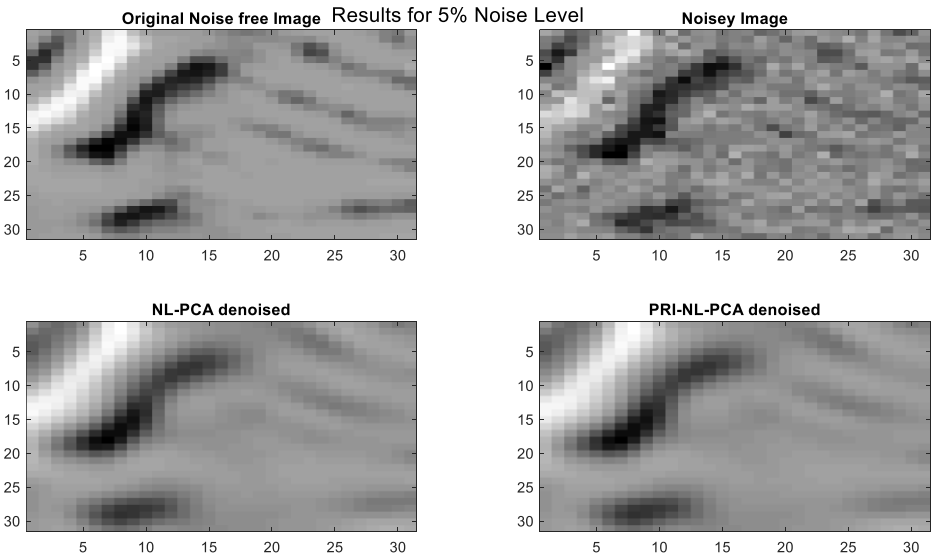
لازم به ذکر است چون این فیلترها جستجوهای مکانی زیادی انجام می‌دهند نسبتا زمانگیر هستند بنابراین برای حذف نویز تنها از بخشی از تصویر استفاده گردید.

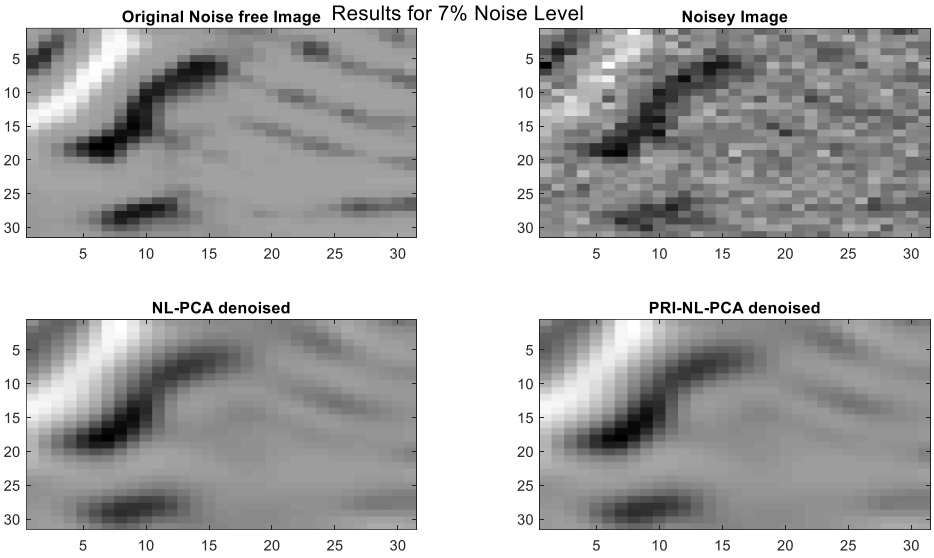
در ادامه فقط به تغییر نوع نویز و مقایسه‌ی معیارهای اعتبار سنجی پرداخته می‌شود.

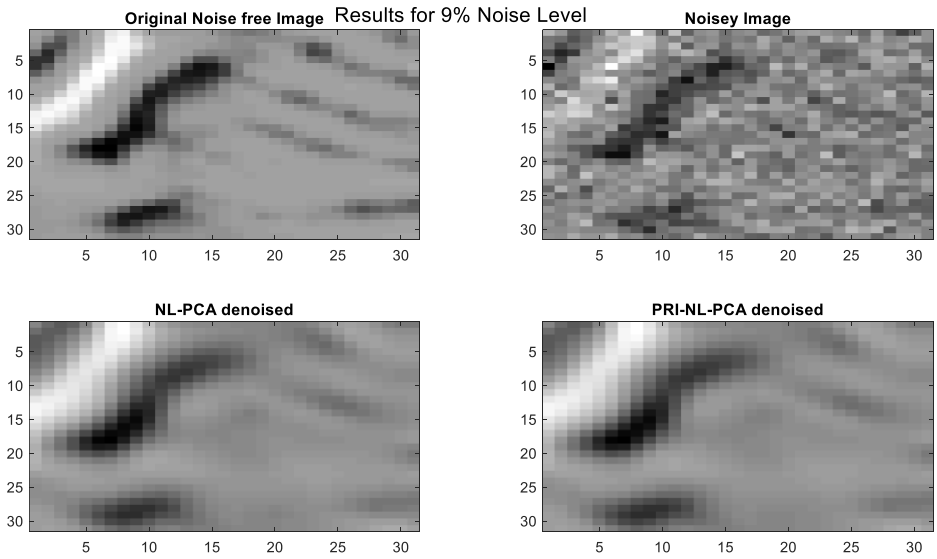
1_2_3 نویز گوسی ثابت

برای کلیه‌ی نویزهای اضافه شونده، نویز با درصدهای 1، 3، 5، 7 و 9 درصد اضافه می‌شود و نتایج نمایش داده می‌شوند. تصاویر تمیز، نویزی، حذف نویز شده با *NL_PCA* و *PRI_NL_PCA* به صورت زیر است.

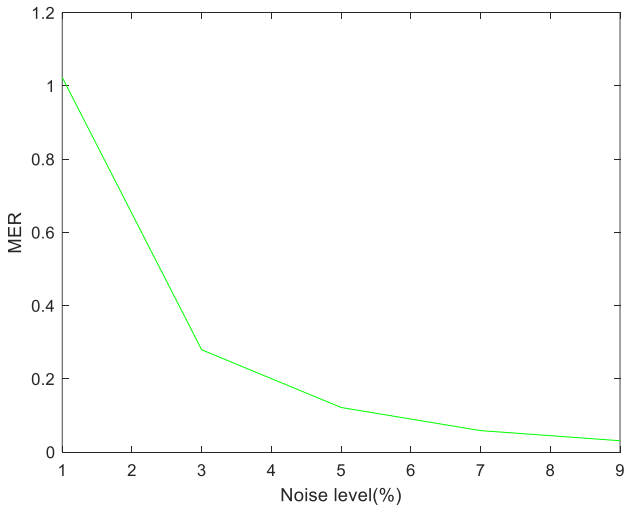



Original Noise free Image
Results for 5% Noise Level
Noisey Image
NL-PCA denoised
PRI-NL-PCA denoised


Original Noise free Image
Results for 7% Noise Level
Noisey Image
NL-PCA denoised
PRI-NL-PCA denoised



معیارهای اعتبار سنجی به صورت زیر است.

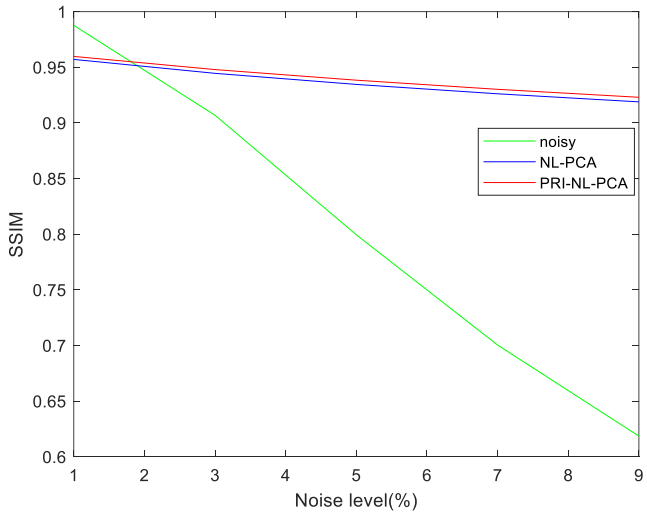


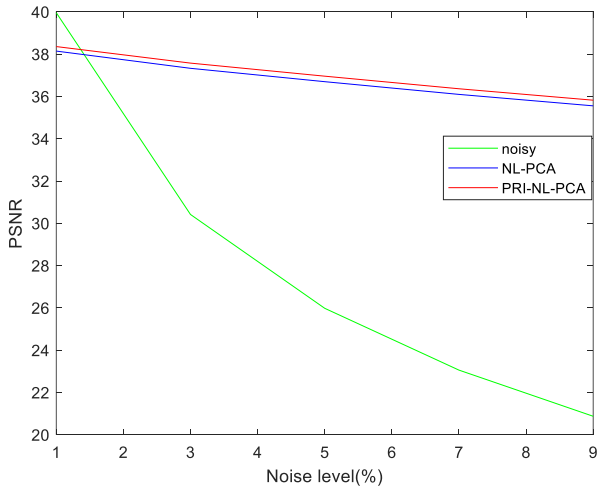
Method	Noise level					
	1%	3%	5%	7%	9%	Average
Median (Eq. 8)	0.2100	0.0715	0.0474	0.0395	0.0367	0.0810
Trimmed median (Eq. 9)	**0.1141**	**0.0490**	**0.0387**	**0.0356**	**0.0345**	**0.0544**

مشاهده می‌شود در نویزهای کوچک الگوریتم نوشته شده نویز را بیشتر از الگوریتم اصلی تخمین می‌زند اما در نویزهای بالاتر دقت آن بسیار بالا است.

Table 9. *PSNR and SSIM results of the compared methods for stationary noise (Gaussian and Rician).*

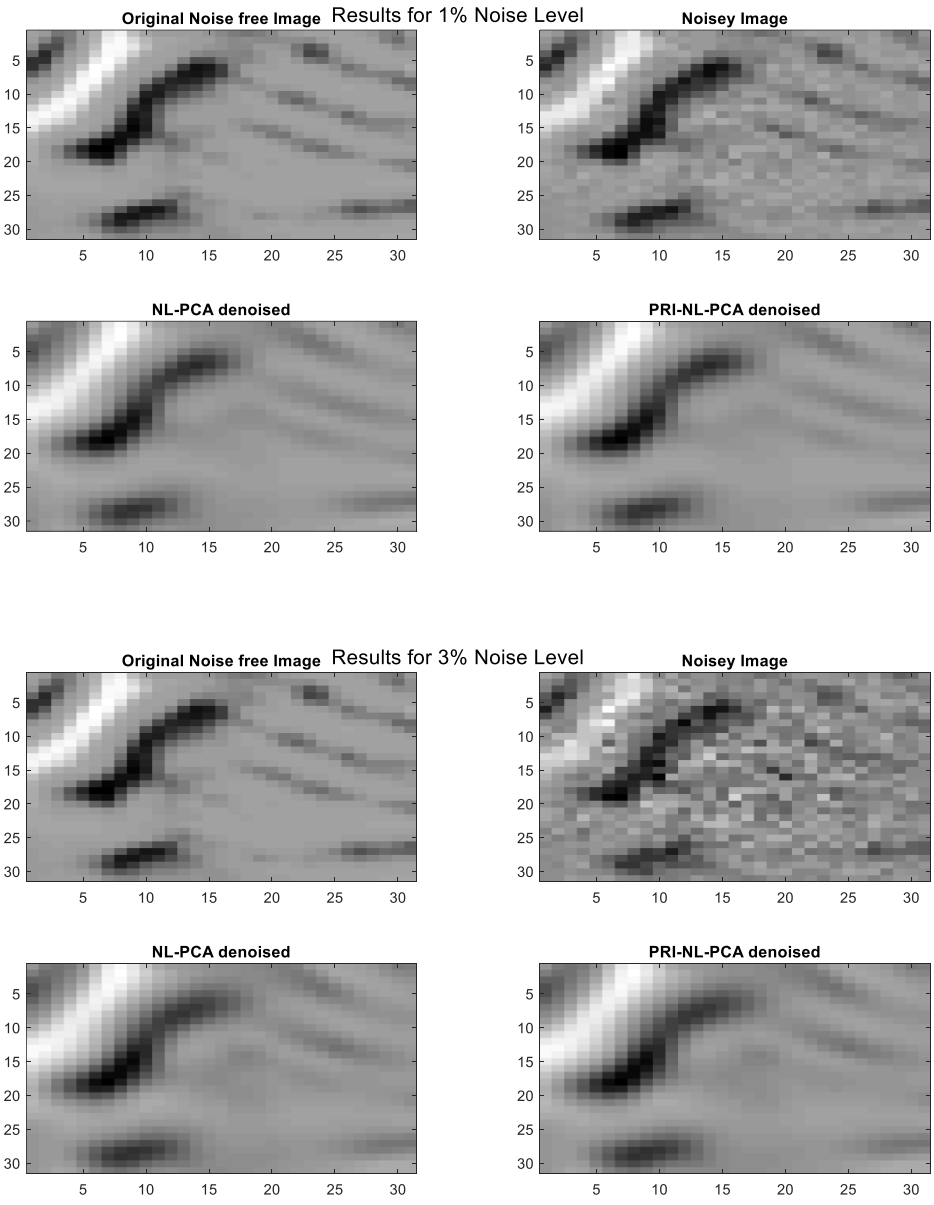
Noise	Filter	Noise level					
		1%	3%	5%	7%	9%	Average
Gauss.	**Noisy**	39.99|0.970	30.46|0.814	26.02|0.656	23.10|0.530	20.91|0.433	28.10|0.681
	ODCT	43.78|0.992	37.53|0.971	34.88|0.951	33.18|0.932	31.91|0.913	36.27|0.952
	PRI-NLM	44.22|0.993	38.34|0.976	35.58|0.959	33.75|0.940	32.37|0.922	36.85|0.958
	ONL-PCA	44.80|0.993	38.93|0.978	36.39|0.964	34.70|0.949	33.39|0.935	37.64|0.964
	NL-PCA	44.80|**0.994**	38.97|0.979	36.40|0.964	34.67|0.948	33.32|0.931	37.63|0.963
	OPRI-NL-PCA	45.20|**0.994**	**39.40|0.981**	**36.69|0.968**	**34.94|0.955**	**33.61|0.941**	**37.97|0.968**
	PRI-NL-PCA	**45.38|0.994**	39.33|**0.981**	36.63|**0.968**	34.90|**0.955**	33.58|**0.941**	37.96|**0.968**
	BM4D	44.02|0.992	38.35|0.975	35.91|0.960	34.31|0.945	33.10|0.930	37.14|0.960

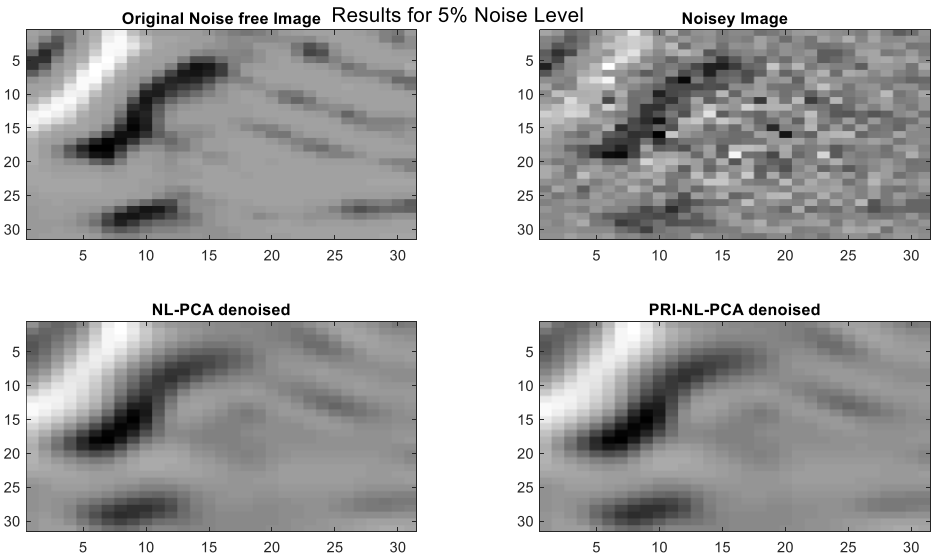


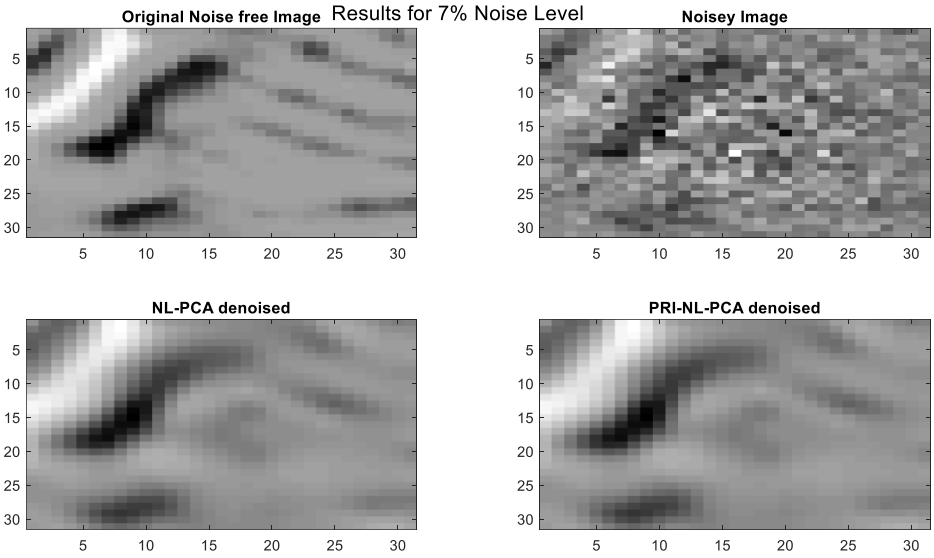


مشاهده می‌شود که نتیجه در نویزهای با شدت بالا تقریبا با الگوریتم اصلی برابر است.

2_2_3 نویز گوسی متغیر با مکان




Original Noise free Image
Results for 5% Noise Level
Noisey Image
NL-PCA denoised
PRI-NL-PCA denoised


Original Noise free Image
Results for 7% Noise Level
Noisey Image
NL-PCA denoised
PRI-NL-PCA denoised

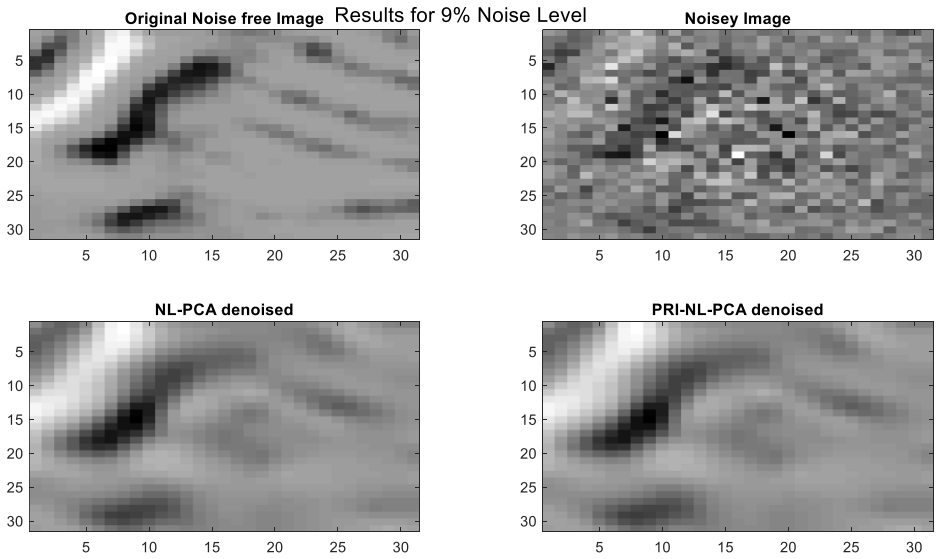


Table 5. *Comparison of the two different noise estimation schemes for non stationary noise (MER)*

Method	Noise level					MER
	1-3%	3-9%	5-15%	7-21%	9-27%	Average
ABM4D	0.2115	0.0715	0.0540	0.0527	0.0549	0.0889
NL-PCA (Eq. 7)	**0.0765**	**0.0409**	**0.0370**	**0.0363**	**0.0365**	**0.0454**

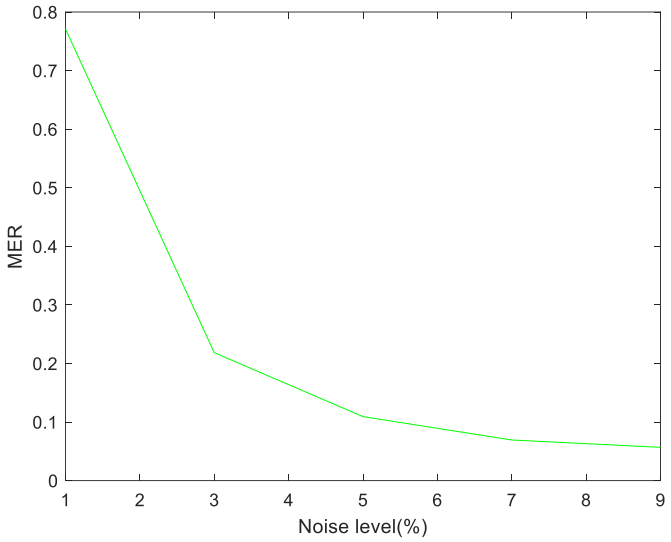
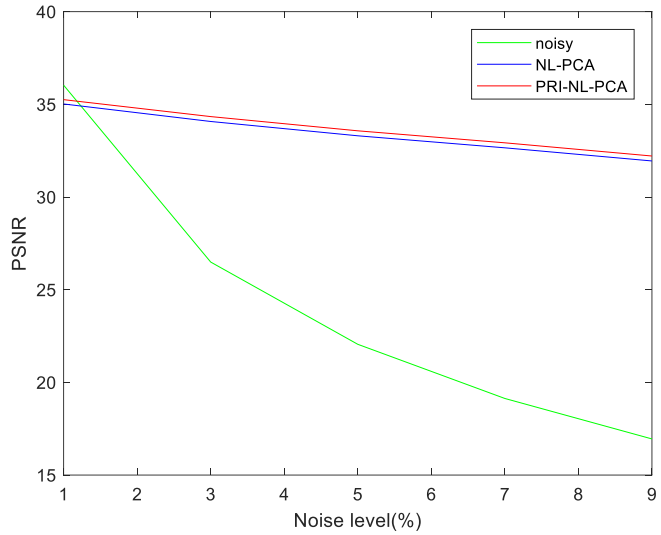
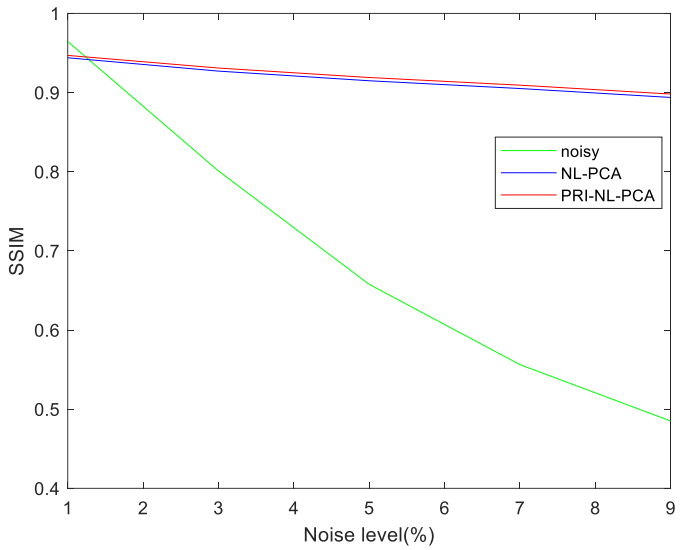


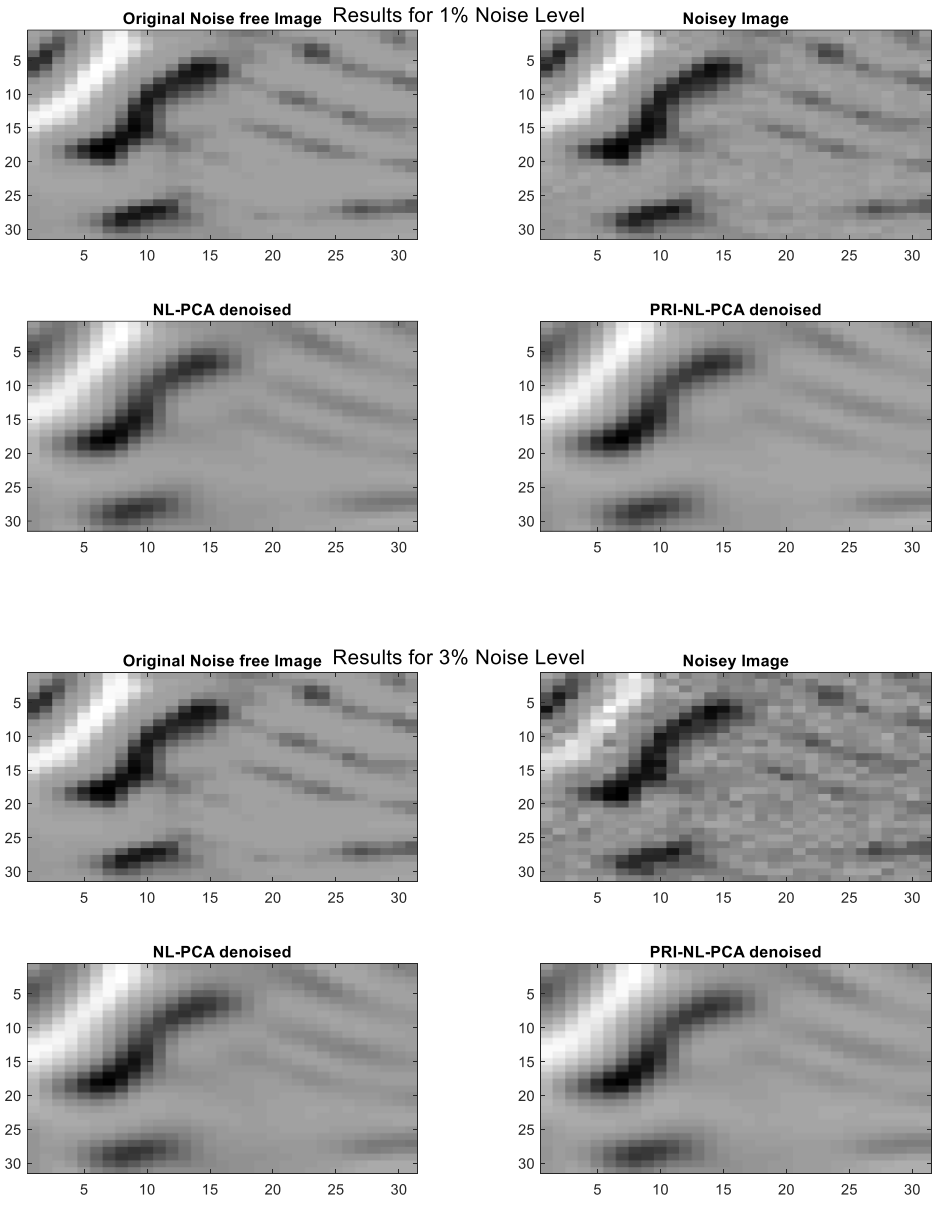
Table 10. *PSNR and SSIM results of the compared methods for spatially varying noise.*

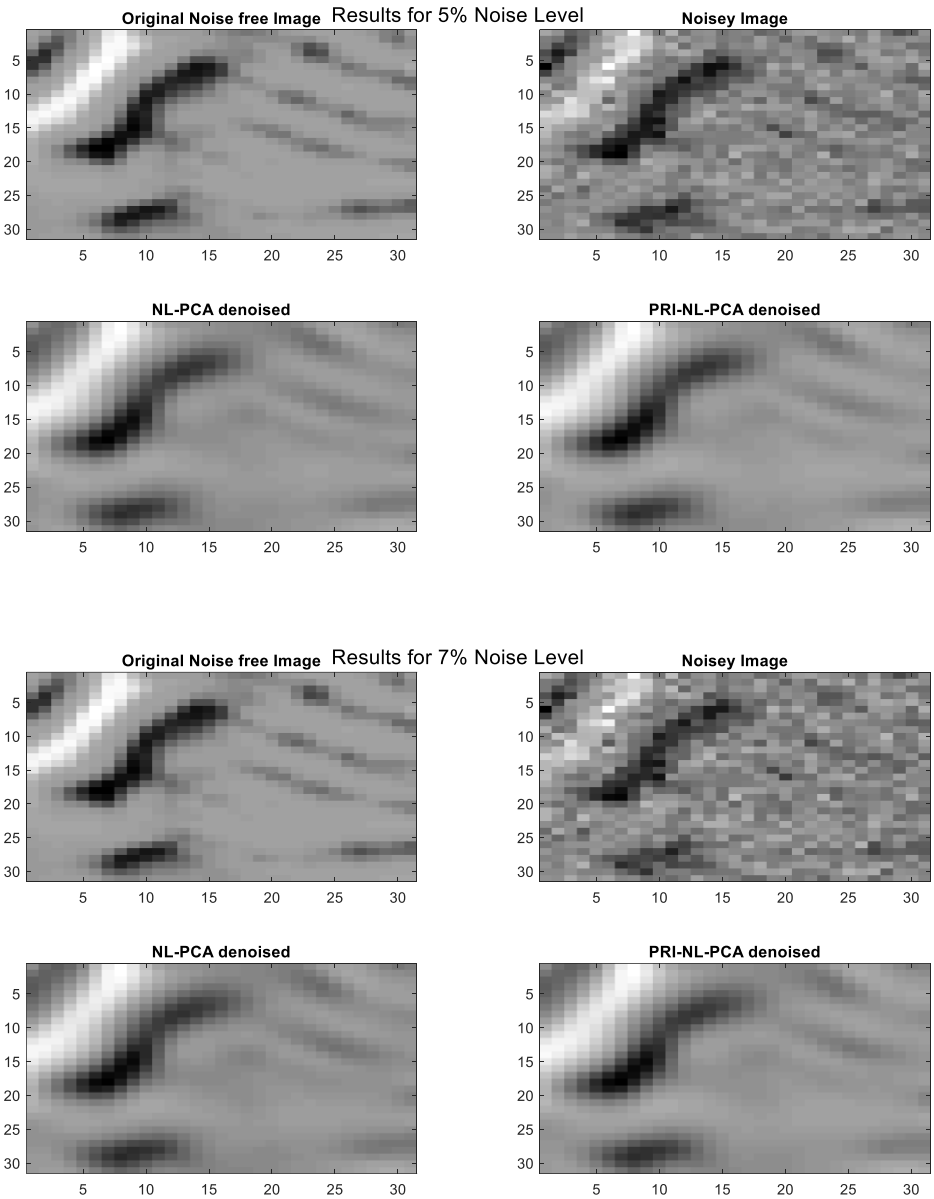
Noise	Filter	Noise Level					
		1-3%	**3-9%**	**5-15%**	**7-21%**	**9-27%**	**Average**
Gauss.	**Noisy**	34.34|0.900	24.80|0.621	20.36|0.442	17.44|0.328	15.26|0.253	22.44|0.508
	ONL-PCA	41.60|0.987	35.94|0.959	33.23|0.928	31.41|0.897	30.07|0.867	34.45|0.928
	NL-PCA	41.66|0.987	35.95|0.958	33.17|0.925	31.31|0.891	29.92|0.857	34.40|0.924
	OPRI-NL-PCA	42.19|**0.989**	**36.33|0.965**	**33.53|0.939**	31.59|**0.911**	30.10|0.882	34.75|**0.937**
	PRI-NL-PCA	**42.25|0.989**	36.30|**0.965**	33.52|**0.939**	**36.61|0.911**	30.16|**0.883**	**34.77|0.937**
	ABM4D	40.45|0.980	35.48|0.960	33.10|0.930	31.48|0.900	**30.24**|0.870	34.15|0.928
	SANLM	40.38|0.980	34.50|0.940	31.57|0.890	29.61|0.830	28.11|0.780	32.83|0.884





3_2_3 نویز ثابت Rician




Original Noise free Image
Results for 5% Noise Level
Noisey Image
NL-PCA denoised
PRI-NL-PCA denoised
Original Noise free Image
Results for 7% Noise Level
Noisey Image
NL-PCA denoised
PRI-NL-PCA denoised

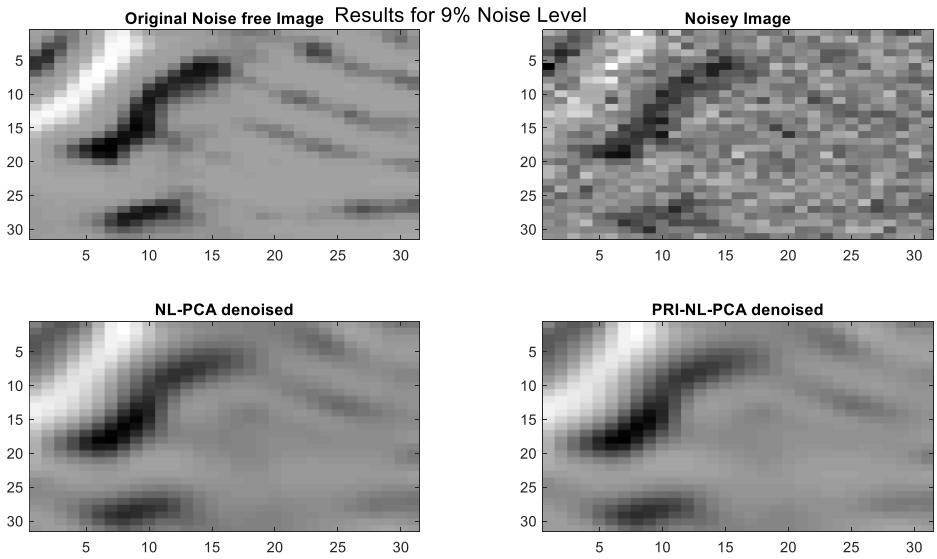
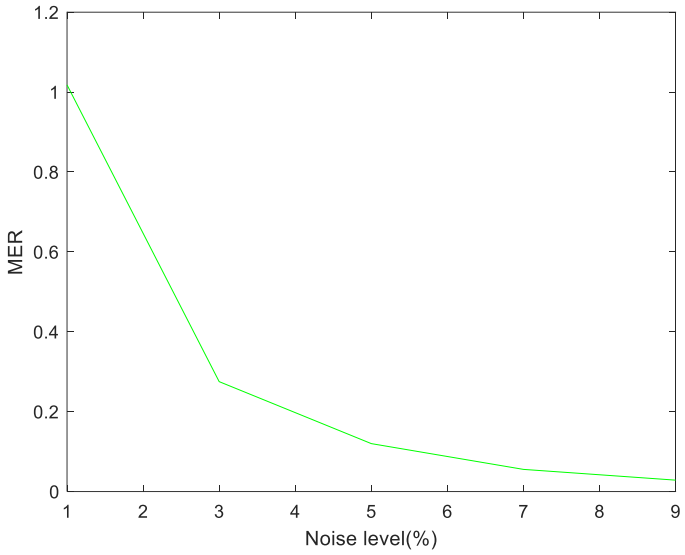
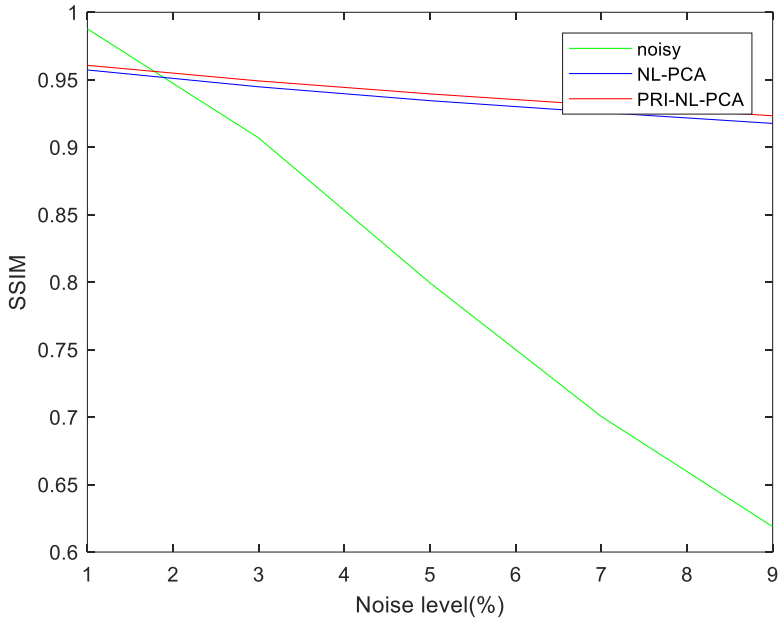


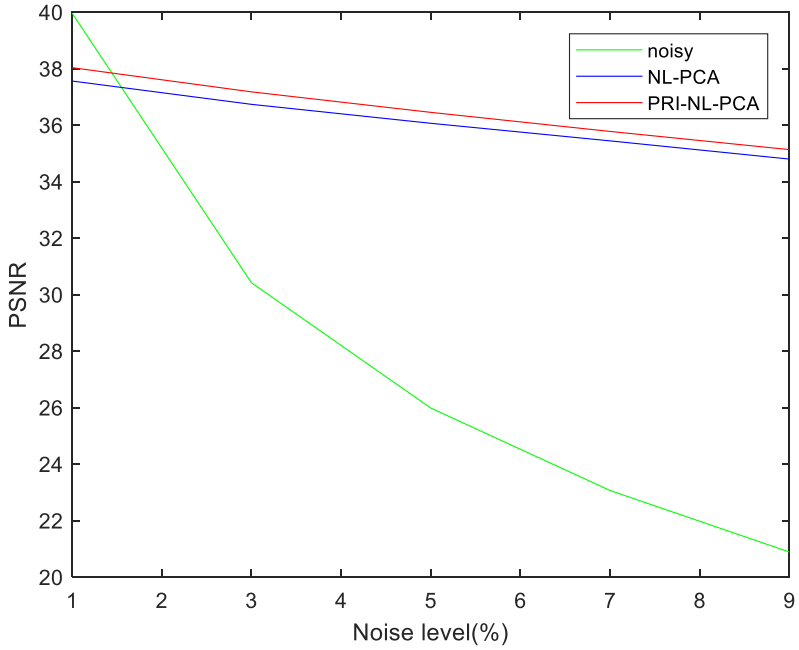
Table 7. *Comparison of the two different noise estimation schemes for different levels of stationary Rician noise (ER).*

Method	Noise level					
	1%	3%	5%	7%	9%	Mean
Mapping residuals (Eq. 11)	**0.1029**	**0.0661**	**0.0514**	**0.0432**	**0.0361**	**0.0599**
ABM4D	0.4965	0.1860	0.1388	0.1315	0.1387	0.2183

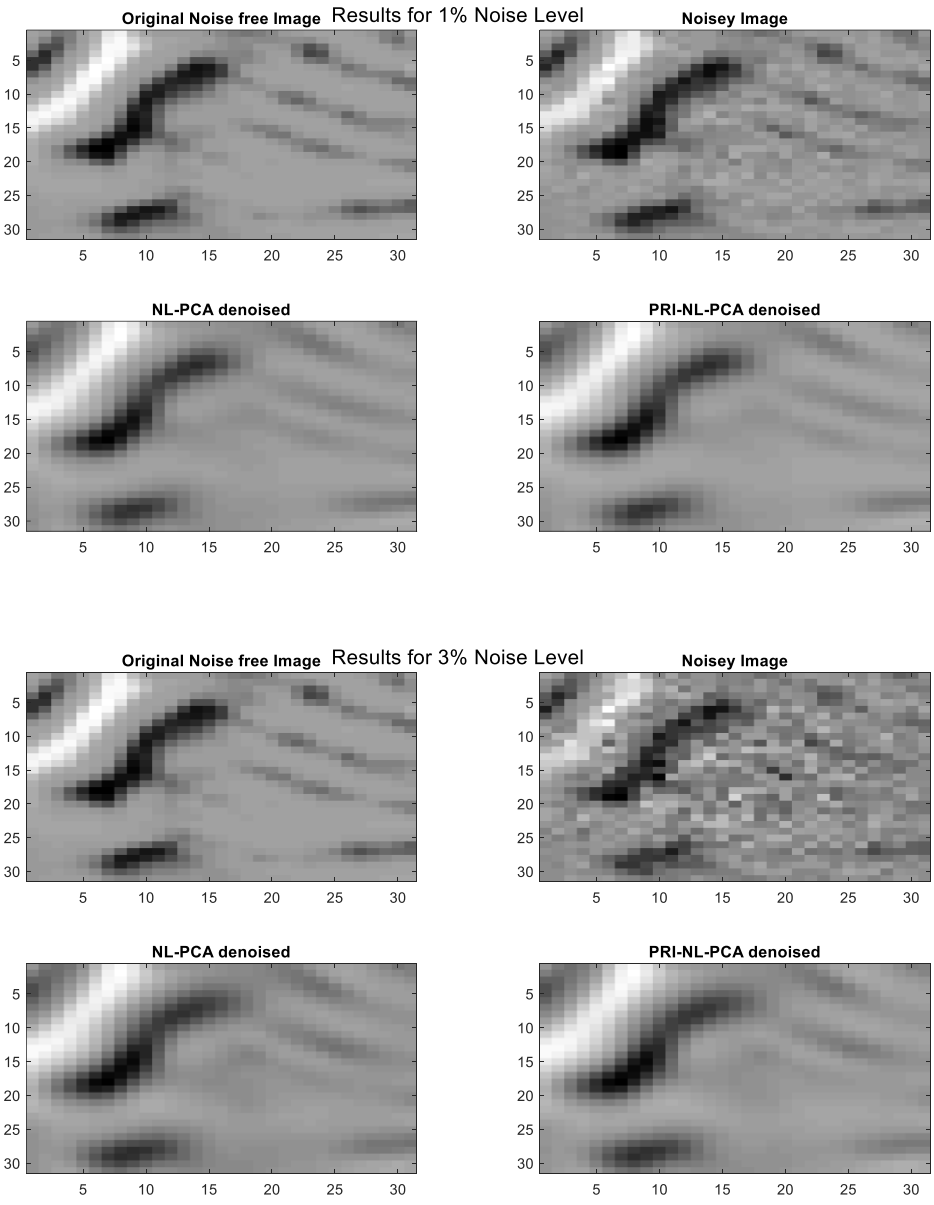


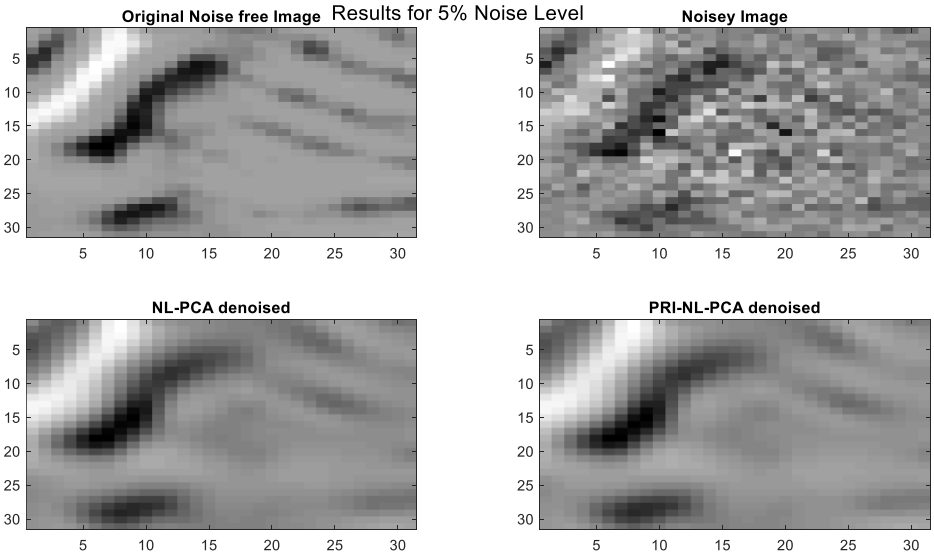
Rician	**Noisy**	40.00|0.970	30.49|0.815	26.09|0656	23.20|0.529	21.04|0.431	28.16|0.680
	ODCT	42.96|0.991	37.38|0.970	34.70|0.949	32.90|0.927	31.53|0.905	35.89|0.948
	PRI-NLM	44.14|0.993	38.28|0.976	35.42|0.957	33.47|0.935	31.98|0.913	36.66|0.955
	ONL-PCA	44.80|0.993	38.89|0.978	36.31|0.963	34.53|0.957	33.11|0.925	37.53|0.962
	NL-PCA	44.79|**0.994**	38.90|0.978	36.23|0.962	34.37|0.943	32.88|0.923	37.43|0.960
	OPRI-NL-PCA	45.20|**0.994**	**39.35|0.981**	**36.59|0.967**	**34.75|0.952**	**33.28|0.935**	37.83|**0.966**
	PRI-NL-PCA	**45.31|0.994**	39.34|**0.981**	36.58|**0.967**	34.74|**0.952**	**33.28|0.935**	**37.85|0.966**
	BM4D	44.09|0.992	38.35|0.975	35.84|0.959	34.17|0.942	32.88|0.924	36.99|0.958

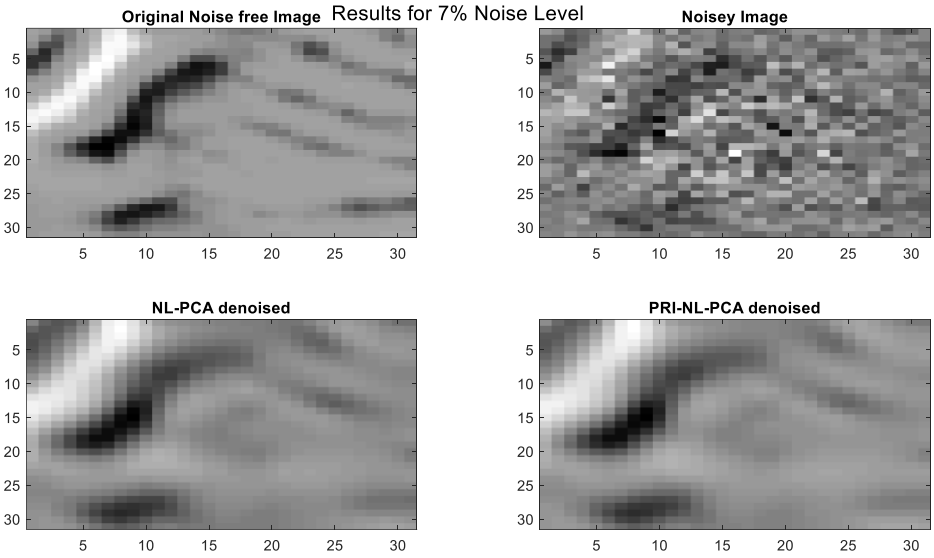




4_2_3 نویز Rician متغیر با مکان




Original Noise free Image
Results for 5% Noise Level
Noisey Image
NL-PCA denoised
PRI-NL-PCA denoised


Original Noise free Image
Results for 7% Noise Level
Noisey Image
NL-PCA denoised
PRI-NL-PCA denoised

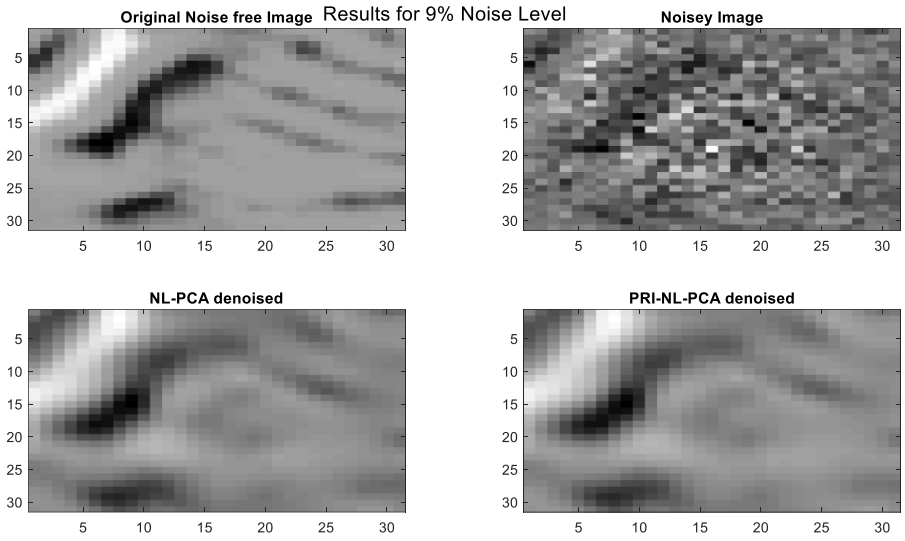
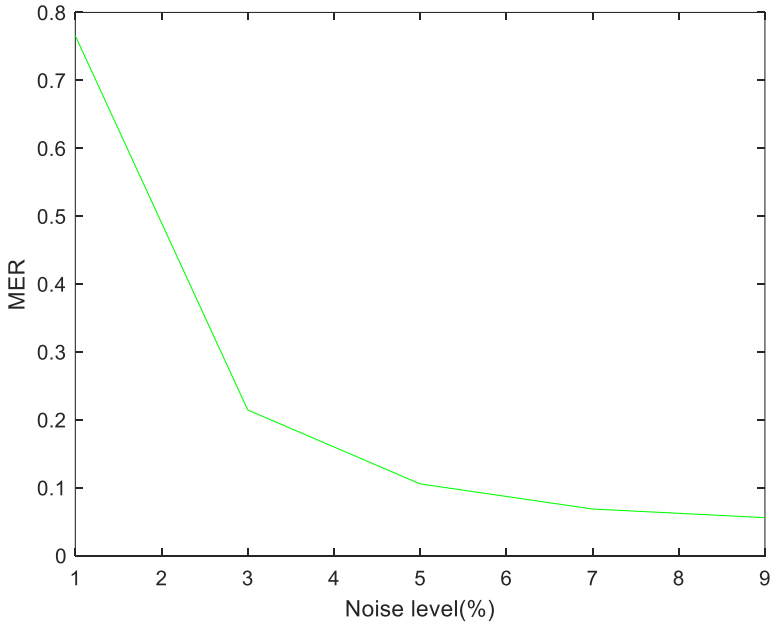
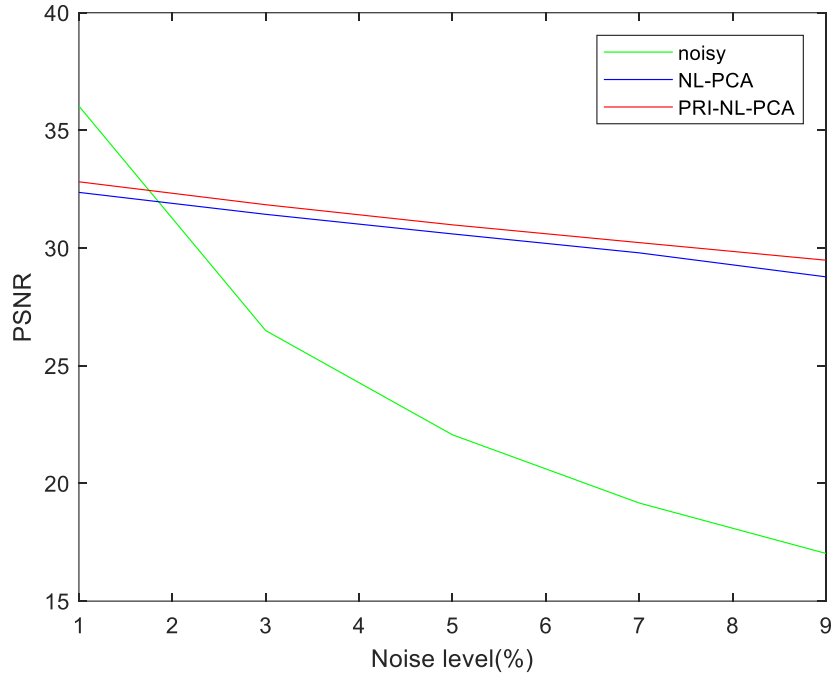


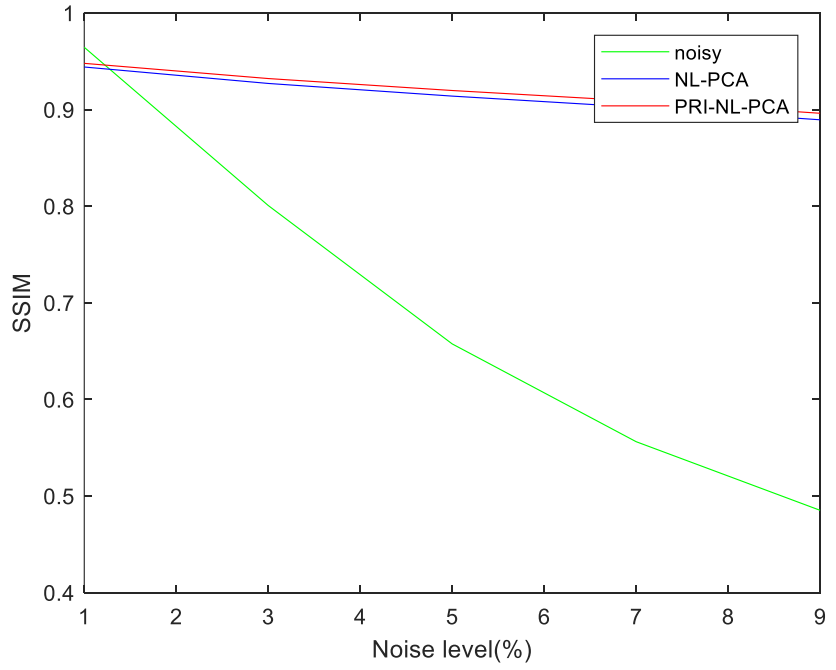
Table 8. *Comparison of the two different noise estimation schemes for different levels of spatially varying Rician noise (MER).*

Method	Noise level					
	1-3%	3-9%	5-15%	7-21%	9-27%	Mean
NL-PCA	**0.0850**	**0.0517**	**0.0400**	**0.0356**	**0.0355**	**0.0496**
ABM4D	0.3524	0.1781	0.1745	0.1868	0.2012	0.2186



Rician	**Noisy**	34.35|0.900	24.87|0.621	20.50|0.441	17.64|0.325	15.50|0.247	22.57|0.507
	ONL-PCA	41.59|0.987	35.87|0.958	32.99|0.925	30.93|0.890	29.28|0.854	34.13|0.923
	NL-PCA	41.64|0.987	35.77|0.956	32.74|0.917	30.48|0.873	28.42|0.824	33.81|0.911
	OPRI-NL-PCA	42.18|**0.989**	**36.20|0.964**	**33.16**|0.933	**30.92|0.897**	**29.22|0.861**	**34.34|0.930**
	PRI-NL-PCA	**42.23|0.989**	36.19|**0.964**	33.15|**0.934**	30.87|**0.897**	28.83|0.856	34.25|0.928
	ABM4D	40.43|0.980	34.41|0.940	31.27|0.890	28.80|0.820	26.55|0.740	32.29|0.874
	SANLM	40.28|0.980	34.29|0.940	31.16|0.870	28.73|0.810	26.43|0.740	31.18|0.868





مشاهده شد که الگوریتم پیاده‌سازی شده نسبت به الگوریتم اصلی در نویزهای با واریانس پایین عملکرد ضعیفتر اما در نویزهای بالاتر عملکرد نسبتا برابر و در بعضی موارد (نویز 7 درصد) عملکرد بهتری نیز داشت.

4 مراجع

[1] Manjón, J.V., P. Coupé, and A. Buades, *MRI noise estimation and denoising using non-local PCA.* Medical image analysis, 2015. **22**(1): p. 35-47 %@ 1361-8415.

[2] Zhang, L., et al., *Two-stage image denoising by principal component analysis with local pixel grouping.* Pattern recognition, 2010. **43**(4): p. 1531-1549 %@ 0031-3203.

[3] Manjón, J.V., et al., *New methods for MRI denoising based on sparseness and self-similarity.* Medical image analysis, 2012. **16**(1): p. 18-27 %@ 1361-8415.

[4] Manjón, J.V., et al., *Diffusion weighted image denoising using overcomplete local PCA.* PloS one, 2013. **8**(9): p. e73021 %@ 1932-6203.