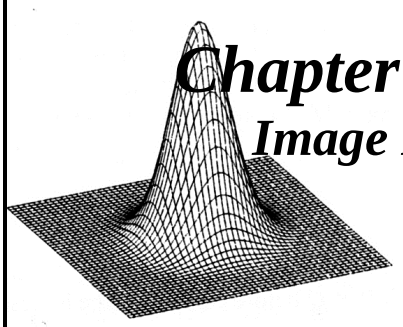


Chapter 5
Image Restoration

بازگرداندن تصویر به حالت اصلی

پیش نمایش(Preview)

هدف از بازگرداندن تصویر به حالت اصلی بهبود آن به شکلی تعریف شده است. گرچه حین بهبود و بازگرداندن تصویر به حالت اصلی برخی قسمتها با یکدیگر همپوشی دارند ولی فرایند اول ذاتی است در حالی که بازگرداندن تصویر به حالت اصلی یک فرایند خنثی است. حین بازگرداندن تصویر به حالت اصلی سعی می‌شود تصویر مستهلک شده بازیابی شود و از اطلاعاتی که حین کاهش رنگ تصویر داشتیم بهره‌برداری می‌کنیم. بنا بر این در شگردهای بازگرداندن تصویر به حالت اصلی ابتدا از روند تنزل کیفی تصویری الگوبرداری می‌شود و برای بازگرداندن تصویر به حالت اصلی فرایند معکوس اعمال می‌شود.

در این روش یک معیار کیفی فرمول‌بندی می‌شود که نتایج مطلوب تخمین زده شود. شگردهای تقویت و بهبود تصویر روشهای مبتنی بر آزمایش و خطا هستند که برای مدیریت کردن تصویر از جنبه‌های روانی – جسمی قوه تجسم انسان بهره می‌برند. مثلاً روش کشش وضوح تصویر شگردی برای تقویت تصویر است و مبتنی بر ارائه جنبه‌های خوشایند تصویر به بیننده است در صورتی که زدودن تاری تصویر با استفاده از ابزار تیره‌زدا شگردی برای بازگرداندن تصویر به حالت اصلی است.

در این فصل نحوه بهره‌برداری از قابلیت‌های نرم افزار MATLAB و IPT برای الگوبرداری از پدیده تنزل کیفی و فرمول‌بندی راه حل‌های بازگرداندن تصویر به حالت اصلی را شرح می‌دهیم. همچون فصلهای ۳ و ۴ می‌بینیم که برخی شگردهای بازگرداندن تصویر به حالت اصلی بهتر است در دامنه فضایی فرمول‌بندی شوند، ولی برخی از آنها برای دامنه فرکانس مناسب هستند.

الگویی از فرایند کاهش رنگ تصویر و بازگرداندن تصویر به حالت اصلی

A Model of Image Degradation /Restoration Process

همان طور که در تصویر ۵.۱ دیده می‌شود فرایند کاهش رنگ به صورت تابع کاهش رنگ در این فصل الگوبرداری شده است. این تابع همراه با عبارت پارازیت انداز روی داده‌های تصویر ورودی $f(x,y)$ عمل می‌کند تا تصویری با رنگهای کاهش یافته$g(x,y)$ تولید کند.

$$g(x,y) = H[f(x,y) + \eta(x,y)]$$

با داشتن اطلاعاتی در خصوص تابع کاهش رنگ H ، و عبارت پارازیت انداز $\eta(x,y)$ ، هدف از بازگرداندن تصویر به حالت اصلی تخمین زدن $f(x,y)$ در تصویر اصلی است. این برآورد باید تا حد امکان به داده‌های تصویر اصلی نزدیک باشد. به طور کلی هر چه اطلاعات بیشتری در خصوص H و η داشته باشیم $\hat{f}(x,y)$ به $f(x,y)$ نزدیکتر خواهد بود.اگر H یک فرایند خطی تغییرناپذیر فضایی(linear Spatially invariant) باشد می‌توان ثابت کرد که تصویر مستهلک شده در دامنه فضایی(spatial domain) ارائه شده است.

$$g(x,y) = h(x,y) * f(x,y) + \eta(x,y)$$

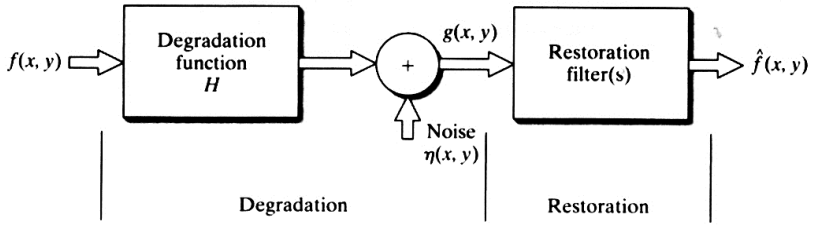
در اینجا $h(x,y)$ نمایش فضایی تابع کاهش رنگ است و نماد ستاره همچون فصل ۴ نمایانگر تلفیق است. در بحث قسمت ۴.۳.۱. گفتیم که تلفیق در دامنه فضایی و ضرب در دامنه فرکانس یک زوج تبدیل فوریر ایجاد می‌کند بنا بر این می‌توان الگوی قبل را به صورت نماد دامنه فرکانس(frequency domain) معادل نوشت:

$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

در این فرمول عبارتهای دارای حروف بزرگ تبدیل فوریر در عبارتهای مطابق با آنها در معادله تلفیق هستند. تابع کاهش رنگ $h(u, v)$ گاهی تابع انتقال نور(Optical transfer function) نیز نامیده می‌شود. این عبارت از تحلیل سیستم‌های نوری در فوریر برگرفته شده است. در دامنه فضایی $h(x, y)$ را تابع انتشار نقطه(Point spread function) می‌نامیم. این عبارت ناشی از عملکرد $h(x, y)$ بر روی یک نقطه نورانی برای به دست آوردن خصوصیات کاهش رنگ برای هر نوع داده‌های ورودی است. تابعهای انتقال نور و انتشار نقطه زوج‌های تبدیل فوریر هستند و این جعبه ابزار ۲ تابع برای تبدیل کردن بین آنها دارد.

از آنجائی که کاهش رنگ به دلیل عملکرد تابع خطی تغییرناپذیر فضایی H حین تلفیق الگوبرداری می‌شود گاهی فرایند کاهش رنگ را تلفیق تصویر با تابع انتقال نور یا تابع انتشار نقطه نیز می‌نامیم و گاهی بازگرداندن تصویر به حالت اصلی را تلفیق‌زدایی(deconvolution) می‌نامیم.در ۳ بخش زیر فرض می‌کنیم که H عملگر هویت نما است و کاهش رنگ را فقط با توجه به پارازیت بررسی می‌نیم. در قسمت ۵.۶ چند روش برای بازگرداندن تصویر به حالت اصلی در حضور h و η بررسی کردیم.

. برای نشان دادن تلفیق از ستاره داخل خط استفاده می‌کنیم و برای نشان دادن مزدوج پیچیده از ستاره بالانویس استفاده می‌کنیم. برای نشان دادن ضرب از ستاره عبارتهای نرم افزار MATLAB استفاده می‌کنیم. باید مواظب باشید که کاربردهای نامرتبط این نماد را با یکدیگر اشتباه نگیرید.



تصویر ۵.۱ الگویی از فرایند بازگرداندن تصویر به حالت اصلی و کاهش رنگ آن

۵.۲. الگوهای پارازیت(Noise Models)

توانایی‌های شبیه سازی رفتار و جلوه‌های پارازیت نقش اصلی در پردازش تصویر دارد. در این فصل به ۲ نوع الگوی پارازیت می‌پردازیم: پارازیت در دامنه فضایی (که با تابع چگالی احتمالی (PDF) پارازیت توصیف می‌شود) و پارازیت در دامنه فرکانس که با خصوصیات گوناگون فوریر پارازیت تشریح می‌شود. به استثنای مطالب بخش ۵.۲.۳. در این فصل فرض می‌کنیم که این پارازیت مستقل از مختصات تصویر است.

۵.۲.۱. افزودن پارازیت با تابع imnoise (Adding Noise with Function imnoise)

در جعبه ابزار برای تحریف کردن تصویر پارازیت‌دار از تابع imnoise استفاده می‌شود. این تابع ترکیب زیر را دارد:

```
g = imnoise(f, type, parameters)
```

در اینجا f تصویر ورودی است. کلمات تایپ و پارامتر را بعداً شرح می‌دهیم. تابع imnoise تصویر ورودی را به نوع double در دامنه [0,1] قبل از افزودن پارازیت به آن تبدیل می‌کند. حین مشخص کردن پارامترهای پارازیت باید این موضوع در نظر گرفته شود مثلاً برای افزودن پارازیت گوسی با میانگین ۶۴ و اختلاف ۴۰۰ به تصویر uint8 میانگین مقیاس را روی 64/255 و اختلاف را روی $400/(255)^2$ در تصویر ورودی imnoise قرار می‌دهیم. شکلهای ترکیبی این تابع به شرح زیر است:

g=imnoise (f, aussian, m, var)_ پارازیت گوسی با میانگین M و اختلاف var را به تصویر f اضافه می‌کند. حالت پیش فرض میانگین پارازیت صفر با اختلاف ۰.۰۱ است.

g=imnoise (f, localvar, v)_ میانگین صفر پارازیت گوسی موضعی با اختلاف وی را به تصویر f اضافه می‌کند. در اینجا وی آرایه‌ای به اندازه f است که حاوی مقادیر اختلاف مطلوب در هر نقطه است.

g=imnoise (f, localvar, image intensity, var)_ میانگین صفر پارازیت گوسی را به تصویر f اضافه می‌کند. در اینجا اختلاف موضعی پارازیت var تابعی از مقادیر شدت رنگ تصویر در f است. شناسه‌های image-intensity, var بردارهای هم اندازه هستند و plot(image-intensity, var) رابطه کاربردی بین اختلاف پارازیت و شدت رنگ تصویر را ترسیم می‌کند. بردار شدت رنگ تصویر باید مقادیر هنجارمند در حیطه [0,1] داشته باشد.

g=imnoise (f,salt & pepper, d)_ تصویر f را با پارازیت انداز افشان که d چگالی پارازیت است تحریف می‌کند. (یعنی این درصدی از مساحت تصویر است که حاوی مقادیر پارازیت‌دار است). بنا بر این تقریباً d*numel-f عدد عنصر تصویری تحت تاثیر قرار می‌گیرند. مقدار پیش فرض چگالی پارازیت ۰.۰۵ است.

_g=imnoise (f, speckle, var) پارازیت تکثیری به تصویر f اضافه می‌کند و از معادله g=f+n*f بهره‌برداری می‌کند در اینجا n پارازیت تصادفی است که یکسان توزیع شده است و میانگین صفر و اختلاف var را دارد. مقدار پیش فرض var ۰.۰۴ است.

_g=imnoise (f, poisson) به جای افزودن پارازیت مصنوعی به داده‌ها پارازیت Poisson را از داده‌ها تولید می‌کند. شدت رنگ تصویرهای uint8, uint16 برای این که با خصوصیات آماری poisson سازگار باشند، باید مطابق با تعداد فتونها (یا سایر مقادیر اطلاعاتی باشند).

وقتی تعداد فتونهای هر عنصر تصویری بیشتر از ۶۵۵۳۵ عدد (ولی کمتر از ۱۰ به توان ۱۲) باشد از تصویرهایی با دقت مضاعف استفاده می‌شود.

مقادیر شدت رنگ بین صفر و ۱ نوسان دارد و مطابق با تعداد فتونهای تقسیم شده بر ۱۰ به توان ۱۲ است.[12](10)

imnoise در بخشهای زیر در چند مثال تشریح شده است.

۵.۲.۲. ایجاد پارازیت تصادفی فضایی با میزان توزیع مشخص

Generating Spatial Random Noise with a spatial Distribution

گاهی وقتها باید پارازیتهایی با انواع و پارامترهایی فراتر از آنچه تابع imnoise می‌سازد ایجاد کنیم.

مقادیر پارازیتهای فضایی تصادفی هستند و یک تابع چگالی احتمالی (Probability density function (PDF دارند که مطابق با تابع توزیع انباشته (Cumulative distribution function (CDF است. تولید تصادفی ارقام در انواع روندهای توزیع مورد نظر ما مطابق با قوانین ساده فرضیه احتمالات است.

در برخی نرم افزارهای تولید ارقام این کار بر حسب ارقام دلخواه با تابع توزیع انباشته (CDF) یکسان در فاصله [0,1] ایجاد می‌شود. در برخی موارد مولد ارقام مورد نظر اعداد گوسی با میانگین صفر و اختلاف واحد ایجاد می‌کند. گرچه می‌توان این دو نوع پارازیت را با استفاده از imnoise ایجاد کرد ولی استفاده از تابع rand در نرم افزار MATLAB برای تولید ارقام دلخواه و randn برای تولید ارقام دلخواه گوسی به کار برده می‌شود. این تابع‌ها بعداً در این فصل تشریح می‌شوند.

شالوده روش تشریح شده در این بخش نتیجه مشخص احتمالات (peebles[1993]) است که طبق آن در صورتی که w یک متغیر دلخواه با توزیع یکسان در فاصله [0,1] باشد در این صورت، می‌توان با حل معادله زیر متغیر دلخواه z را با تابع توزیع انباشته (CDF), F_z به دست آورد.

$$Z=F_z^{-1}(w)$$

این نتیجه قاطع را می‌توان با عنوان یافتن راه حلی برای معادله $F_z(z)=w$ ذکر کرد.

مثال ۵.۱ :فرض کنید مولد w را برای تولید ارقام دلخواه در فاصله [0,1] داریم، و فرض کنید می‌خواهیم آن را برای تولید ارقام دلخواه z با تابع توزیع انباشته (CDF) به شکل زیر ایجاد کنیم:

$$F_z(z)=\begin{cases}1-e^{-(z-a)^2/b} & for \quad z\geq a\\ 0 & for \quad z<a\end{cases}$$

در اینجا b>0 است و برای پیدا کردن z معادله را حل می‌کنیم.

$$1-e^{-(z-a)^2/b}=w$$
$$z=a+\sqrt{-b\ln(1-w)}$$

از آنجائی که عبارت ریشه جذر غیرمنفی است فرض می‌کنیم مقادیر z کمتر از a تولید نمی‌شوند. این یکی از شرط‌های تعریف تابع توزیع انباشته (CDF) توسط Rayleigh است. بنا بر این رقم دلخواه w از مولد قبل در معادله قبل برای تولید متغیر دلخواه z با توزیع Rayleigh و پارامترهای a, b استفاده می‌شود.

این نتیجه در نرم افزار MATLAB به آرایه آر با ارقام دلخواه با عبارت زیر تعمیم داده می‌شود:

```
>> R = a + sqrt(-b*log(1 – ran(M, N));
```

در اینجا همان طور که در بخش ۳.۲.۲ گفتیم log لگاریتم طبیعی است و همان طور که قبلاً گفتیم rand ارقام توزیع شده یکسان در فاصله [0,1] ایجاد می‌کند. اگر فرض کنیم M=N=1 باشند در این صورت، خط فرمان نرم افزار MATLAB مقداری با متغیر دلخواه با روند توزیع Rayleigh با خصوصیات پارامترهای a, b ایجاد می‌کند.

گاهی عبارت $z=a+\sqrt{-b\ln(1-w)}$ را معادله مولد ارقام دلخواه می‌نامیم زیرا نحوه ایجاد ارقام دلخواه مطلوب را مشخص می‌کند. در این مورد توانستیم یک راه حل نزدیک پیدا کنیم. ولی این کار همیشه امکان‌پذیر نیست و باید معادله مولد ارقام دلخواه را پیدا کنیم که داده‌های خروجی آن نزدیک به ارقام دلخواه با تابع توزیع انباشته (CDF) مشخص باشند.

در جدول ۵.۱ متغیرهای دلخواه مورد نظر در این مبحث همراه با تابع چگالی احتمالات و تابع توزیع انباشته (CDF) و معادله‌های مولد ارقام دلخواه فهرست بندی شده‌اند. در برخی موارد همچون متغیرهای نمایی و Rayleigh می‌توان راه حلی برای تابع توزیع انباشته (CDF) و معکوس آن پیدا کرد. بنا بر این می‌توان عبارتی را برای مولد ارقام دلخواه بر حسب ارقام دلخواه تغییرناپذیر نوشت که در مثال ۵.۱ تشریح شده است. در سایر موارد مثل چگالیهای گوسی و lognormal راه حل نزدیک برای تابع توزیع انباشته (CDF) وجود ندارد و باید برای تولید ارقام دلخواه روشهای دیگری پیدا کرد. مثلاً در lognormal متغیر دلخواه z طوری است که in(z) توزیع گوسی دارد و عبارت جدول ۵.۱ بر حسب متغیرهای دلخواه گوسی با میانگین صفر و اختلاف واحد نوشته می‌شود. در سایر موارد می‌توان برای به دست

آوردن راه حلی ساده‌تر این مسئله را مجددا فرمول‌بندی کرد. مثلاً ارقام دلخواه Erlang با پارامترهای a, b با افزودن ارقام دلخواه با توزیع نمایی b که پارامتر a دارند به دست می‌آید.

مولدهای ارقام دلخواه در imnoise و جدول ۵.۱ نقش مهمی در الگوبرداری رفتار پارازیتهای دلخواه در برنامه‌های پردازش تصویر دارند. مزایای توزیع یکسان و تولید ارقام دلخواه با تابع‌های توزیع انباشته گوناگون را قبلاً دیدیم. پارازیت گوسی برای نزدیک کردن مقادیر در مواردی همچون حسگرهای تصویربردار که با نور پایین کار می‌کنند به کار برده می‌شود. پارازیت افشان (Pepper noise) در وسایل معیوب انتقال دهنده به کار می‌رود. اندازه ذرات نقره در لایه حساس تصویر یک متغیر دلخواه است که با توزیع lognormal توصیف می‌شود. ریشه پارازیت Rayleigh در تصویربرداری این حیطه است. ولی پارازیتهای Erlang و نمایی برای توصیف پارازیت در تصویربرداری لیزری به کار برده می‌شوند.

تابع imnoise2 که در آخر این بخش شرح داده شده است ارقام دلخواهی ایجاد می‌کند که در جدول ۵.۱ تابع توزیع انباشته (CDF) دارند. این تابع از تابع rand در نرم افزار MATLAB استفاده می‌کند و در این فصل ترکیب زیر را دارد:

```
A = rand(M, N)
```

TABLE 5.1 Generation of random variables.

Name	PDF	Mean and Variance	CDF	Generator[†]
Uniform	$p_z(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \le z \le b \\ 0 & \text{otherwise} \end{cases}$	$m = \frac{a+b}{2}, \quad \sigma^2 = \frac{(b-a)^2}{12}$	$F_z(z) = \begin{cases} 0 & z < a \\ \frac{z-a}{b-a} & a \le z \le b \\ 1 & z > b \end{cases}$	MATLAB function rand
Gaussian	$p_z(z) = \frac{1}{\sqrt{2\pi}b} e^{-(z-a)^2/2b^2}$ $-\infty < z < \infty$	$m = a, \quad \sigma^2 = b^2$	$F_z(z) = \int_{-\infty}^{z} p_z(v)\, dv$	MATLAB function randn
Salt & Pepper	$p_z(z) = \begin{cases} P_a & \text{for } z = a \\ P_b & \text{for } z = b \\ 0 & \text{otherwise} \end{cases}$ $b > a$	$m = aP_a + bP_b$ $\sigma^2 = (a-m)^2 P_a + (b-m)^2 P_b$	$F_z(z) = \begin{cases} 0 & \text{for } z < a \\ P_a & \text{for } a \le z < b \\ P_a + P_b & \text{for } b \le z \end{cases}$	MATLAB function rand with some additional logic
Lognormal	$p_z(z) = \frac{1}{\sqrt{2\pi}bz} e^{-[\ln(z)-a]^2/2b^2}$ $z > 0$	$m = e^{a+(b^2/2)}, \sigma^2 = [e^{b^2} - 1]e^{2a+b^2}$	$F_z(z) = \int_0^z p_z(v)\, dv$	$z = ae^{bN(0,1)}$
Rayleigh	$p_z(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & z \ge a \\ 0 & z < a \end{cases}$	$m = a + \sqrt{\pi b/4}, \sigma^2 = \frac{b(4-\pi)}{4}$	$F_z(z) = \begin{cases} 1 - e^{-(z-a)^2/b} & z \ge a \\ 0 & z < a \end{cases}$	$z = a + \sqrt{-b \ln[1 - U(0,1)]}$
Exponential	$p_z(z) = \begin{cases} ae^{-az} & z \ge 0 \\ 0 & z < 0 \end{cases}$	$m = \frac{1}{a}, \quad \sigma^2 = \frac{1}{a^2}$	$F_z(z) = \begin{cases} 1 - e^{-az} & z \ge 0 \\ 0 & z < 0 \end{cases}$	$z = -\frac{1}{a} \ln[1 - U(0,1)]$
Erlang	$p_z(z) = \frac{a^b z^{b-1}}{(b-1)!} e^{-az}$ $z \ge 0$	$m = \frac{b}{a}, \quad \sigma^2 = \frac{b}{a^2}$	$F_z(z) = \left[1 - e^{-az} \sum_{n=0}^{b-1} \frac{(az)^n}{n!}\right]$ $z \ge 0$	$z = E_1 + E_2 + \cdots + E_b$ (The E's are exponential random numbers with parameter a.)

[†] $N(0, 1)$ denotes normal (Gaussian) random numbers with mean 0 and a variance of 1. $U(0, 1)$ denotes uniform random numbers in the range $(0, 1)$.

این تابع ارایه‌ای به اندازه M*N تولید می‌کند که ورودی‌های آن ارقامی با توزیع یکسان هستند و مقادیر آنها در بازه [0,1] است. اگر N حذف شود مقدار پیش فرض آن M می‌شود. اگر بدون شناسه فراخوانی شود، rand عدد دلخواهی ایجاد می‌کند که هر دفعه تابع فراخوانی می‌شود تغییر می‌کند. بنا بر این تابع زیر

```
A = randn(M, N)
```

آرایه‌ای به اندازه M*Nتولید می‌کند که عناصر آن اعداد گوسی عادی با میانگین صفر و اختلاف واحد هستند. اگرN حذف شود مقدار پیش فرض آنM می‌شود. اگر بدون شناسه فراخوانی شود randn یک رقم دلخواه تولید می‌کند.

تابع imnoise2 از تابع find نرم افزار MATLAB استفاده می‌کند که به شکل ترکیبی زیر است:

```
I = find(A)
[r, c]=find(A)
[r, c, v]=find(A)
```

شکل اول کلیه شاخصهای آرایهA را در I تولید می‌کند که دلالت بر عناصر غیرصفر دارند. اگر هیچ کدام از آنها پیدا نشوند، تابع find ماتریس خالی را تولید می‌کند. شکل دوم شاخص‌های ستون و ردیف ورودیهای غیرصفر را در ماتریسA ایجاد می‌کند. شکل سوم غیر از تولید شاخص‌های ردیف و ستون مقادیر غیرصفر a را به صورت بردار ستونی v تولید می‌کند.

در شکل اول آرایهA به شکل A(:) آماده می‌شود بنا بر این I بردار ستون است. این شکل در پردازش تصویر خیلی مفید است. مثلاً برای پیدا کردن و تنظیم کلیه عناصر یک تصویر که مقادیر آنها کمتر از ۱۲۸ است به کار برده می‌شود.

```
>> I = find(A < 128);
>> A(I) = 0;
```

عبارت منطقی A<128 برای آن دسته از عناصر A که این شرط منطقی را داشته باشند مقدار ۱ و برای آن دسته که این شرط را ندارند مقدار ۰ را تولید می‌کند. برای قرار دادن کلیه عناصر تصویری در فاصله [64,192] عبارت زیر را می‌نویسیم:

```
>> I = find(A >= 64 & A <= 192);
>> A(I) = 128;
```

تابع M زیر بر خلاف imnoise آرایه پارازیت R را به اندازه M*N تولید می‌کند که به هیچ وجه مقیاس بندی نمی‌شود. تفاوت مهم دیگر آن است که imnoise یک تصویر پارازیت‌دار در داده‌های خروجی خود تولید می‌کند در حالی که imnoise2 الگوی پارازیت را تولید می‌کند.

کاربر مقادیر مشخص پارامتر پارازیت را خودش تعیین می‌کند. توجه داشته باشید که آرایه پارازیت ناشی از پارازیت افشان (Pepper noise) سه مقدار ممکن است داشته باشد: صفر مطابق با پارازیت افشان (Pepper noise)، ۱ مطابق با پارازیت افشان سفید و ۰.۵ مطابق با حالت بدون پارازیت.

برای مفید کردن این ارایه باید بیشتر پردازش شود. مثلاً برای تحریف کردن یک تصویر با این آرایه با استفاده از تابع find کلیه مشخصات R را با مقدار صفر پیدا می‌کنیم و مختصات مطابق با آن را در تصویر روی کمترین مقدار امکان‌پذیر مقیاس خاکستری (که معمولاً صفر است) قرار می‌دهیم. سپس کلیه مختصات R را که مقدار ۱ دارند پیدا می‌کنیم و کلیه مختصات تصویر را روی حداکثر مقدار ممکن (که در یک تصویر ۸ بیت معمولاً ۲۵۵ است) تنظیم می‌کنیم . در این فرایند مشخص می‌شود که چگونه پارازیت افشان (Pepper noise) بر تصویر تاثیر می‌گذارد.

```
function R = imnoise2(type, M, N, a, b)
%IMNOISE2 Generates an array of random numbers with specified PDF.
%     R = IMNOISE2 (TYPE, M, N, A, B) generates an array, R, of size
%     M-by-N, whose elements are random numbers of the specified TYPE
%     with parameters A and B. If only TYPE is included in the
%     input argument list, a single random number of the specified
%     TYPE and default parameters shown below is generated. If only
%     TYPE, M, and N are provided, the default parameters shown below
%     are used. If M = N = 1, IMNOISE2 generates a single random
%     number of the specified TYPE and parameters A and B.
%
%     Valid values for TYPE and parameters A and B are:
%
%     'uniform'   Uniform random numbers in the interval (A, B).
%                 The default values are (0, 1).
%
%     'gaussian'  gaussian random numbers with mean A and standard
%                 deviation B. The default values are A = 0, B = 1.
%
%     'salt & pepper'   salt and pepper numbers of amplitude 0 with
%                       probability Pa = A, and amplitude 1 with
%                       probability Pa = B. The defult values are Pa =
%                       Pb = A = B = 0.05. Note that the noise has
%                       values 0 (with probability Pa = A) and 1 (with
%                       probability Pa = B), so scaling is necessary if
%                       values other than 0 and 1 are required. The noise
%                       matrix R is assigned three values. If R(x, y) =
%                       0, the noise at (x, y) is pepper (black). If
%                       R(x, y) = 1, the noise at (x, y) is salt
%                       (white). If R(x, y) = 0.5, there is no noise
%                       assigned to coordinates (x, y).
%     'lognormal' Lognormal number with offset A and shape
%                 parameter B. The defaults are A = 1 and B =
%                 0.25.
%     'rayleigh'  Rayleigh noise with parameters A and B. The
%                 default values are A = 0 and B = 1.
%     'exponential'     exponential random numbers with parameters A. The
%                 default is A = 1.
%     'erlang'    Erlang (gamma) random numbers with parameters A
%                 and B. B must be a positive integer. The
%                 defaults are A = 2 and B = 5. Erlang random
%                 numbers are approximated as the sum of B
%                 exponential random numbers.
% Set default values.
if nargin == 1
      a = 0; b = 1;
      M = 1; N = 1;
elseif nargin == 3
```

```
      a = 0; b = 1;
end

% Begin processing. Use lower(type) to protect against input
% begin capitalized.
Switch lower(type)
case 'uniform'
      R = a + (b – a)*rand(M, N);
case 'gaussian'
      R = a + b*rand(M, N);
case 'salt & pepper'
      if nargin <= 3
            a = 0.05; b = 0.05;
      end
      % Check to make sure that Pa + Pb is not > 1.
      if (a + b) > 1
            error('The sum Pa + Pb must not exceed 1.')
      end
      R(1:M, 1:N) = 0.5;
      % Generate an M-by-N array of uniformly-distributed random numbers
      % in the range (0, 1). Then, Pa*(M*N) of them will have values <=
      % a. The coordinates of these points we call 0 (pepper
      % noise). Similarly, Pb*(M*N) points will have values in the range
      % > a & <= (a + b). These we call 1 (salt noise).
      X = rand(M, N);
      C = find(X <= a);
      R(c) = 0;
      u = a + b;
      c = find(X > a & X <= u);
      R(c) = 1;
case 'lognormal'
      if nargin <= 3
            a = 1; b = 0.25;
      end
      R = a*exp(b*randn(M, N));
case 'exponential'
      if nargin <= 3
            a = 1;
      end
      if a <= 0
            error('parameter a must be positive for exponential type.')
      end
      k = -1/a;
      R = k*log(1-rand(M, N));
Case 'erlang'
      if nargin <= 3
            a = 2; b = 5;
      end
      if (b ~= round(b) | b <= 0)
            error('param b must be a positive integer for Erlang.')
      end
      k = -1/a;
      R = zeros(M, N);
      for j = 1:b
            R = R + k*log(1 – rand(M, N));
      end
otherwise
      error('Unknown distribution type.')
end
```

مثال ۵.۲: پیشینه نما(histogram)ی داده‌های تولید شده با استفاده از تابع imnoise2: :

در تصویر ۵.۲. پیشینه نما(histogram)ی کلیه ارقام دلخواه در جدول ۵.۱. نشان داده شده است. داده‌های هر یک از تصویرها با استفاده از تابع imnoise2 تولید شده است. مثلاً داده‌های تصویر ۵.۲.(a) با استفاده از فرمان زیر تولید شدند:

```
>> r = imnoise2('gaussian', 100000, 1, 0, 1);
```

در این عبارت بردار ستونی r با ۱۰۰۰۰۰ عنصر که هر کدام از آنها رقمی دلخواه از روند توزیع گوسی با میانگین صفر و انحراف معیار ۱ هستند تولید شده است. سپس پیشینه نما(histogram)(histogram) با استفاده از تابع hist که ترکیب زیر را دارد به دست آمد.

```
p = hist(r, bins)
```

در اینجا bins تعداد محفظه‌ها است. برای تولید پیشینه نما(histogram) تصویر ۵.۲. bins = 50 در نظر گرفته شد. سایر سابقه نماها به شکلی مشابه تولید شدند. در هر مورد مقادیر انتخاب شده مقادیر پیش فرض بودند، که در تشریح تابع imnoise2 فهرست بندی شده بودند.

۵.۲.۳. پارازیت تناوبی(Periodic Noise)

ریشه پارازیت تناوبی تصویر تداخلهای برقی یا الکترومغناطیس حین تصویربرداری هستند. این تنها پارازیت وابسته به فضا است که در این فصل بررسی شده است. همان طور که در بخش ۵.۴ گفتیم برای ایجاد پارازیت تناوبی در یک تصویر دامنه فرکانس فیلتر می‌شود. الگوی پارازیت تناوبی ما یک الگوی منحنی شکل ۲ بعدی با معادله زیر است:

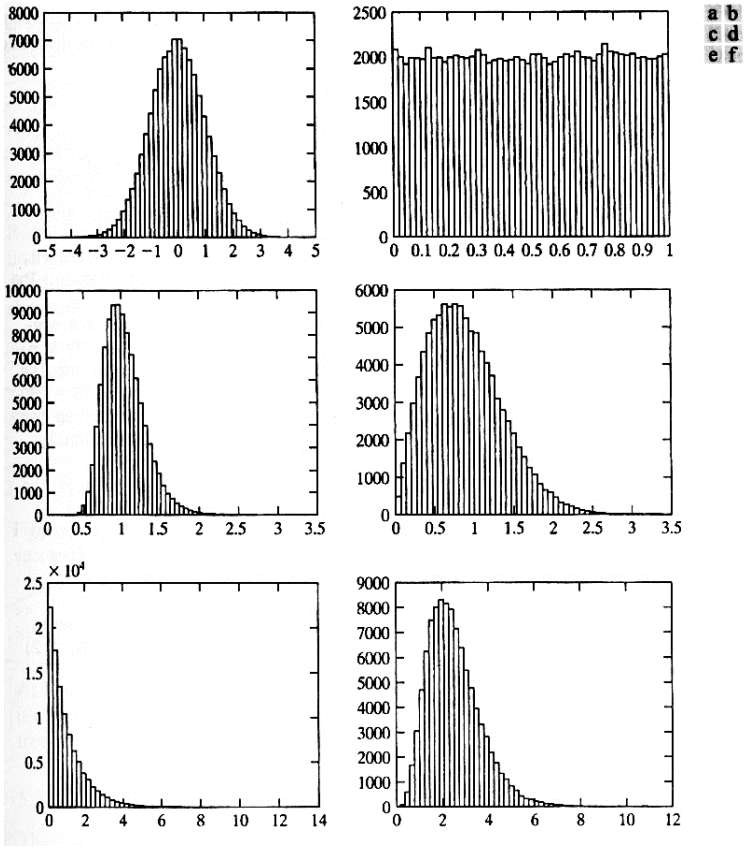
$$r(x,y) = A\sin[2\pi u0(x+B_x)/M + 2\pi v_0(y+B_y)/N]$$

در اینجا a میزان بزرگی، هستند و u0, v0 فرکانس انحنا را به ترتیب با توجه به محورهای ایکس و ایگرگ مشخص می‌کنند و B_x,B_y جابجایی مرحله‌ای با توجه به منشاء هستند. تبدیل فوریر M*N این معادله به شرح زیر است:

$$R(u,v) = j\frac{A}{2}[(e^{j2\pi u_0 B_x/M})\delta(u+u_0, v+v_0) - (e^{j2\pi v_0 B_y/M})\delta(u-u_0, v-v_0)]$$

این یک مجموعه تکانه‌های مزدوج پیچیده است که به ترتیب در مواضع زیر قرار دارند:

$(u+u_0 , v+v_0)$, $(u-u_0 , v-v_0)$



تابع M زیر مقدار دلخواه مواضع تکانه (مختصات فرکانس) را می‌پذیرد که هر کدام از آنها دامنه، فرکانس، پارامترهای جابجایی مرحله‌ای خاص خود را دارد و $r(x, y)$ را به صورت منحنی‌هایی که در پاراگراف قبل تشریح شد محاسبه می‌کند.

این تابع مجموع تبدیلهای منحنی فوریر $R(u, v)$ و طیف آن را در داده‌های خروجی قرار می‌دهد. امواج منحنی شکل با توجه به اطلاعات موضع ضربه از طریق تبدیل فوریر معکوس تولید می‌شوند. این امر باعث می‌شود که این کار به طور خودکار انجام شود و تجسم محتوای فرکانس در الگوی پارازیت فضایی ساده می‌شود. برای تعریف کردن موضع ضربه فقط به یک جفت مختصات نیاز داریم.

این برنامه ضربه‌های متقارن مزدوج تولید می‌کند.

(به کاربرد تابع iftshift برای تبدیل R در کانون به اطلاعات صحیح در عملیات ifft2 همان طور که در بخش ۴.۲ گفتیم توجه کنید.

```
function [r, R, S] = imnoise3(M, N, C, A, B)
% IMNOISE3 Generates periodic noise.
%     [r, R, S] = IMNOISE3(M, N, C, A, B), generates a spatial
%     sinusoidal noise pattern , r, of size M-by-N, its Fourier
%     transform, R, and spectrum, S. The remaining parameters are:
%
%     C is a K-by-2 matrix with K pairs of freq. domain coordinates (u,
%     v) that define the locations of impulses in the freq. domain. The
%     locations are with respect to the frequency rectangle center at
%     (floor(M/2) + 1, floor(N/2) + 1). The impulse locations are spe-
%     cified as increments with respect to the center. For ex, if M =
%     N = 512, then the center is at (257, 257). To specify an impulse
%     at (280, 300) we specify the pair (23, 43); i.e., 257 + 23 = 280,
```

```
%     and 257 + 43 = 300. Only one pair of coordinates is required for
%     each impulse. The conjugate pairs are generated automatically.
%
%     A is a 1-by-k vector that contains the amplitude of each of the
%     K impulse pairs. If A is not included in the argument, the
%     default used is A = ONES(1, K). B is then automatically set to
%     its default values (see next paragraph). The value specified
%     for A(j) is associated with the coordinates in C(j, 1:2).
%
%     B is a k-by-2 matrix containing the Bx and By phase components
%     for each impulse pair. The default values for B are B(1:K, 1:2)
%     = 0.
% Process input parameters.
[k, n] = size(C);
if nargin == 3
      A(1:k) = 1.0;
      B(1:k, 1:2) = 0;
Elseif nargin == 4
      B(1:k, 1:2) = 0
End

% Generate R.
R = zeros(M, N);
for j = 1:k
      u1 = M/2 + 1 + C(j, 1); v1 = N/2 + 1 + C(j, 2);
      R(u1, v1) = i * (A(j)/2) * exp(i*2*pi*C(j, 1) * B(j, 1)/M);
      % Complex conjugate.
      u2 = M/2 + 1 - C(j, 1); v2 = N/2 + 1 - C(j, 2);
      R(u2, v2) = -i * (A(j)/2) * exp(i*2*pi*C(j, 2) * B(j, 2)/N);
end

% Compute spectrum and spatial sinusoidal pattern.
S = abs(R);
r = real(ifft2(ifftshift(R)));
```

طیف و الگوی پارازیتهای فضایی با استفاده از فرمانهای زیر در تصویر ۵.۳.(a) و (b) نشان داده شده است.

```
>> C = [0 64; 0 128; 32 32; 64 0; 128 0; -32 32];
>> [r, R, S] = imnoise3(512, 512, C);
>> imshow(S, [ ])
>> figure, imshow(r, [ ])
```

به خاطر داشته باشید که ترتیب مختصات (u, v) است. این دو مقدار با توجه به کانون چهارچوب فرکانس مشخص شده‌اند (برای تعریف مختصات این نقطه کانونی به بخش ۴.۲ مراجعه کنید). نتایج تکرار فرمانهای قبل در تصاویر ۵.۳.(c) و d نشان داده شده است.

```
>> C = [0 32; 0 64; 16 16; 32 0; 64 0; -16 16];
```

در تصویر ۵.۳.(e) رابطه زیر حاصل شد:

```
>> C = [6 32; -2 2];
```

تصویر ۵.۳. f با همان (C) تولید شده است ولی از بردار دامنه غیرپیش فرض استفاده شده است.

```
>> A = [1 5];
>> [r, R, S] = imnoise3(512, 512, C, A);
```

همان طور که در تصویر ۵.۳. f می‌بینید، امواج منحنی کم فرکانس بر تصویر مسلط هستند. همین موضوع نیز انتظار می‌رفت زیرا دامنه آن پنج برابر دامنه مولفه‌های فرکانس بالا است.

۵.۲.۴. تخمین پارامترهای پارازیت(Estimating Noise Parameters)

با تحلیل طیف فوریر در این تصویر پارامترهای پارازیت تناوبی مشخص می‌شوند. پارازیت تناوبی معمولاً نوسانهای فرکانسی کوتاه ایجاد می‌کند که آنها را می‌توان با چشم غیرمسلح نیز دید. تحلیل خودکار در موقعیت‌هایی امکان‌پذیر است که نوسانها کافی باشند و یا در خصوص فرکانس تداخل اطلاعات کافی داشته باشیم.

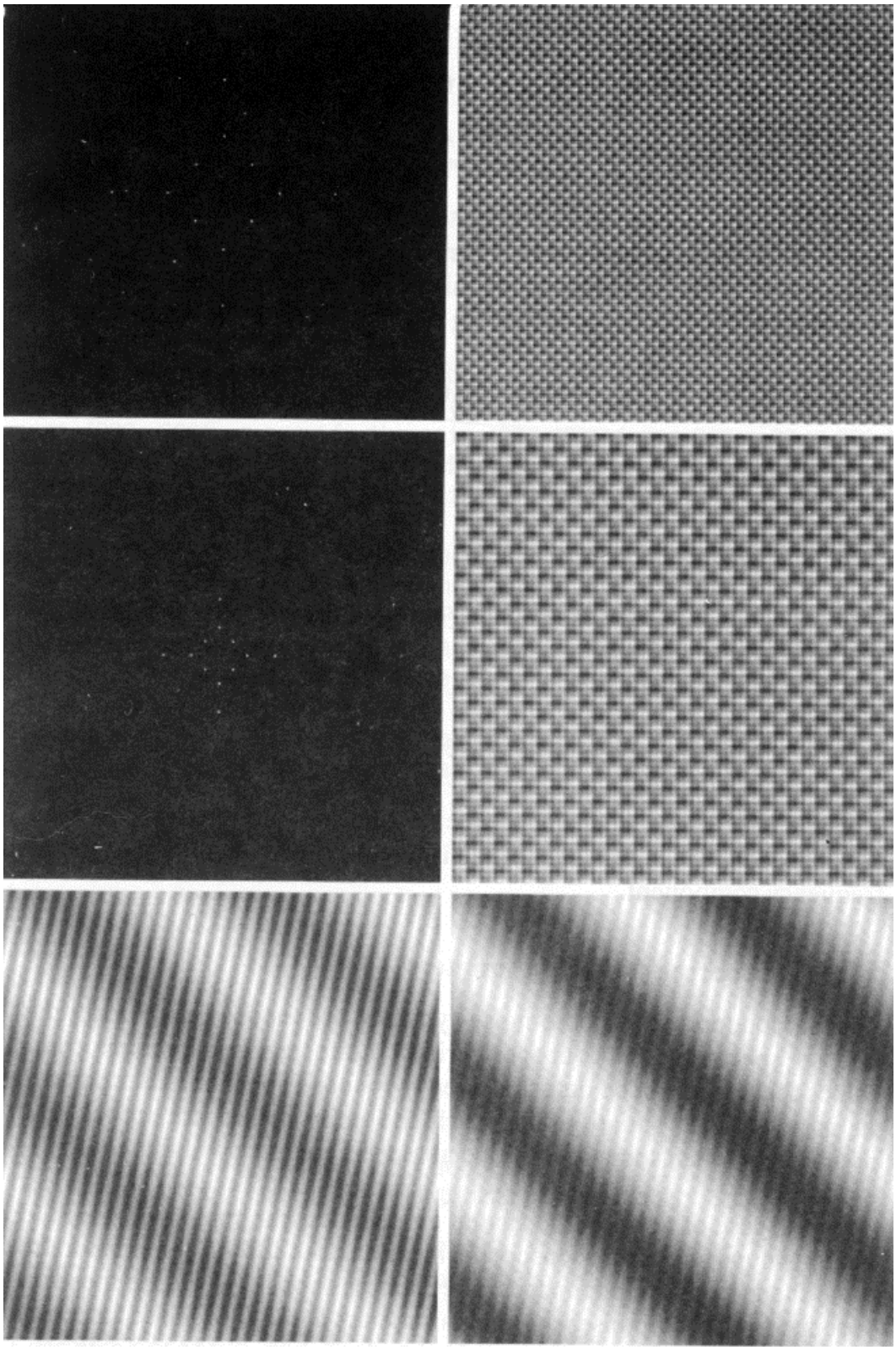
پارامترهای تابع چگالی احتمالات در پارازیتهای دامنه فضایی با مشخصات حسگرها تا حدی مشخص می‌شوند ولی گاهی باید آنها را از نمونه تصویرها تخمین زد. رابطه بین میانگین M و اختلاف δ^2 در پارازیت و پارامترهای a, b که برای مشخص کردن پارازیت تابع چگالی احتمال به کار می‌روند در جدول ۵.۱ فهرست بندی شده است.

بنا بر این مسئله برآورد میانگین و اختلاف با توجه به نمونه تصویرها و استفاده از این تخمینها برای حل a, b است.

فرض کنید zi یک متغیر دلخواه متمایز است که شدت رنگ را در یک تصویر نشان می‌دهد و i=0,1,2,…,L-1 پیشینه نما(histogram) مطابق با آن است.

در اینجا L تعداد مقادیر تشدید امکان‌پذیر است. هر یک از مولفه‌های $p(z_i)$ پیشینه نما(histogram) برآوردی از احتمال وقوع موارد تشدید z_j است.

این پیشینه نما(histogram) می‌تواند مقداری نزدیک به شدت تابع چگالی احتمال(PDF) باشد.



تصویر ۵.۳. ((a)) طیف ضربه‌های مشخص (b) الگوی منحنی مطابق با آن (c)ترتیب مشابه (d) ۲ الگوی پارازیت‌دار دیگر (e) موارد f و پ بزرگنمایی شده‌اند تا دیدن آنها راحت باشد.

یکی از روشهای اصلی برای تشریح شکل پیشینه نما(histogram) از طریق گشتاورهای کانونی است که آنها را گشتاورهای پیرامون میانگین نیز می‌نامند و به شکل زیر تعریف می‌شوند:

$$\mu_n = \sum_{i=0}^{L-1} (z_i - m)^n \, p(z_i)$$

در اینجا n گشتاور ترتیب و M میانگین است:

$$m = \sum_{i=0}^{L-1} z_i \, p(z_i)$$

از آنجائی که پیشینه نما(histogram)(histogram) هنجارمند فرض شده است مجموع کلیه مولفه‌های آن ۱ است بنا بر این از معادله‌های قبل به این رابطه پی می‌بریم. دومین گشتاور

$$\mu_2 = \sum_{i=0}^{L-1} (z_i - m)^2 \, p(z_i)$$

میزان اختلاف است. در این فصل فقط به میانگین و اختلاف توجه می‌کنیم..

تابع statmoments میانگین و گشتاورهای کانونی را تا رده n حساب می‌کند و آنها را در بردار ردیف v تولید می‌کند. از آنجائی که گشتاور رده صفر همیشه ۱ است و گشتاور رده ۱ همیشه صفر است تابع فوق از این ۲ گشتاور صرف نظر می‌کند و در عوض رابطه زیر را در نظر می‌گیرد. $V(1)=m \ ,v(k)=\mu_k \ \text{for} \ k=2,3,\ldots,n$ این ترکیب به شکل زیر است.

```
[v, unv] = statmoments(p, n)
```

در اینجا p بردار پیشینه نما(histogram)(histogram) و n تعداد گشتاورهای محاسباتی است. باید تعداد مولفه‌های p در تصویرهای نوع uint8 ۲ به توان ۸(2^8) و در uint16 ۲ به توان ۱۶(2^{16})، و در حالت مضاعف ۲ به توان ۸ یا ۱۶ باشد. بردار خروجی v حاوی گشتاور هنجارمند مبتنی بر مقادیر متغیر تصادفی است که در حیطه [0,1] مقیاس بندی شده‌اند بنا بر این کلیه گشتاورها در این حیطه هستند. بردار unv همان گشتاورهای v را دارد ولی داده‌های آن در حیطه اصلی محاسبه می‌شوند. مثلاً اگر طول (p)=256 و v(1)=0.5 باشد در این صورت، unv(1) مقدار ۱۲۷.۵ دارد که نیمی از حیطه [0,255]است.

معمولاً پارامترهای پارازیت مستقیماً از یک تصویر یا مجموعه تصویرهای پارازیت‌دار تخمین زده می‌شوند. در این روش هدف انتخاب ناحیه‌ای در یک تصویر است که خصوصیاتی در پس زمینه نداشته باشد طوری که تغییرپذیری مقادیر شدت رنگ در این ناحیه اصولاً به دلیل پارازیت باشد. برای انتخاب ناحیه مورد نظر در نرم افزار MATLAB

از تابع roipoly استفاده می‌کنیم که یک (ROI) چند ضلعی ایجاد می‌کند. این تابع ترکیب زیر را دارد:

```
B = roipoly(f, c, r)
```

در اینجا f تصویر مورد نظر است. و c و r بردارهای مطابق با مختصات ستون و ردیف راس‌های چندضلعی هستند (توجه داشته باشید که ستونها اول قید شده‌اند) داده‌های خروجی B تصویر دودویی به همان اندازه f هستند که صفرها خارج از ناحیه مورد نظر قرار گرفته‌اند و یکها داخل ناحیه هستند. تصویر B همچون نقابی است که عملیات ناحیه مورد نظر را محدود کرده است.

برای مشخص کردن (ROI) چند ضلعی از ترکیب زیر استفاده می‌کنیم:

```
B = roipoly(f)
```

که تصویر f را روی صفحه نشان می‌دهد و کاربر می‌تواند چند ضلعی را با ماوس مشخص کند. اگر f حذف شود، roipoly روی آخرین تصویر نمایش داده شده عمل می‌کند و با چند کلیک می‌توان راسها را به چند ضلعی افزود. فشردن دکمه‌های backspace, delete باعث حذف راسهای انتخاب شده می‌شود. با چپ کلیک یا راست کلیک می‌توان راس نهایی را به ناحیه منتخب اضافه کرد و چند ضلعی با ارقام ۱ پر می‌شود. با زدن دکمه ورود return انتخاب بدون افزودن راس تمام می‌شود.

برای به دست آوردن تصویر دودویی و فهرستی از راسهای چند ضلعی از سازه زیر استفاده می‌کنیم:

```
[B, c, t] = roipoly(. . .)
```

در اینجا roipoly ترکیبهای معتبر این تابع را نشان می‌دهد و cو r مختصات ستون و ردیف راسها هستند. این قالب وقتی که (ROI) محاوره‌ای به کار رود مفید است چون که مختصات راسهای چند ضلعی مورد استفاده در برنامه‌های دیگر و تکثیر (ROI) را نشان می‌دهد.

تابع زیر پیشینه نما(histogram) تصویر را در ناحیه‌ای محاسبه می‌کند که راسهای آن با بردارهای cو r مشخص می‌شوند. به استفاده از تابع roipoly در این برنامه برای تکثیر ناحیه چند ضلعی تعریف شده با cو r توجه کنید.

```
function [p, npix] = histroi(f, c, r)
%HISTROI Computes the histogram of an ROI in an image.
%     [P, NPIX] = HISTROI(F, C, R) computes the histogram, P, of a
%     polygonal region of interest (ROI) in image F.  The polygonal
%     region is defined by the column and row coordinates of its
%     vertices, which are specified (sequentially) in vectors C and R,
%     respectively. All pixels of F must be >=0. Parameter NPIX is the
%     number of pixels in the polygonal region.

% Generate the binary mask image.
B = roipoly(f, c, r);
% Compute the histogram of the pixels in the ROI.
P = imhist(f(B));
% Obtain the number of pixels in the ROI if requested in the output.
if nargout > 1
      npix = sum(B(:));
end
```

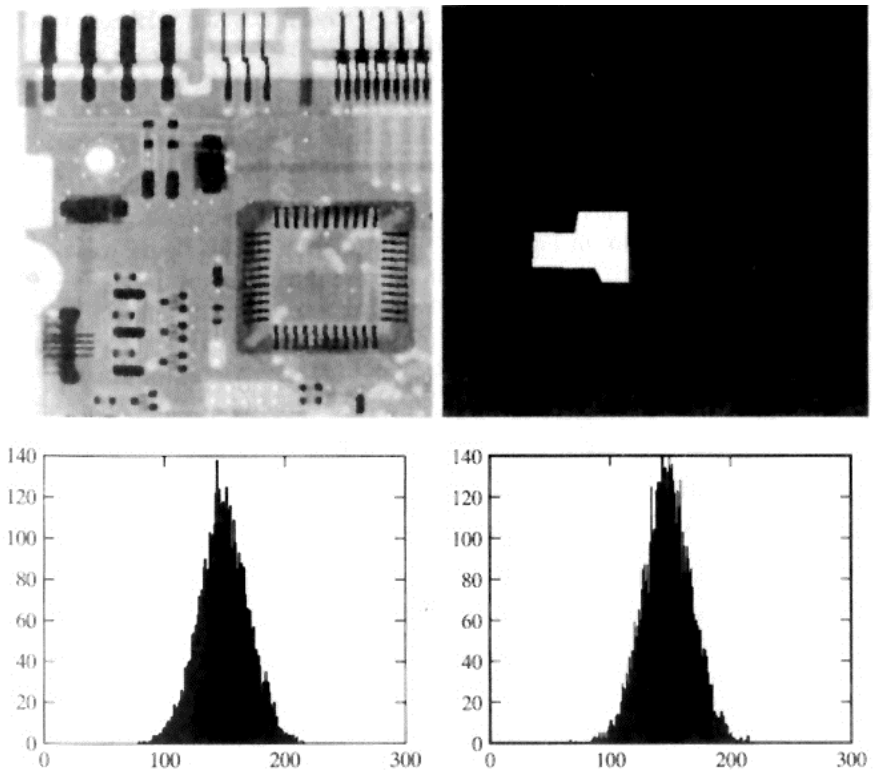
مثال ۵.۴ برآورد پارامترهای پارازیت

یک تصویر پارازیت‌دار در (a)۵.۴ نشان داده شده است که در مبحث زیر با f مشخص شده است. هدف از این مثال برآورد نوع پارازیت و پارامترهای آن با استفاده از شگردها و ابزار ابداع شده تا حالا است. در تصویر (b)۵.۴ نقاب B را که با فرمان زیر به صورت محاوره‌ای تولید شده است می‌بینید.

```
>> [B, c, r] = roipoly(f);
```

تصویر (c)۵.۴ با فرمانهای زیر تولید شد:

```
>> [p, npix] = histroi(f, c, r);
>> figure, bar(p, 1)
```



تصویر ۵.۴. (a)) تصویر پارازیت‌دار (ROI) (b) تولید شده در فرایند محاوره‌ای (c) پیشینه نما(histogram) (ROI) پیشینه نما(histogram) داده‌های گوسی تولید شده با استفاده از تابع imnoise2

میانگین و اختلاف ناحیه نقاب گذاری شده با b به شکل زیر به دست آمد:

```
>>[v, nuv] = statmoments(p, 2);
>> v
v =
      0.5794      0.0063.
>> unv
      147.7430    410.9313
```

در تصویر ۵.۴.(c) مشخص است که این پارازیت تقریباً گوسی است. دانستن میانگین و اختلاف دقیق پارازیت امکان‌پذیر نیست چون که به مق۵یاس خاکستری تصویر در ناحیه ب اضافه شده است. ولی با انتخاب ناحیه‌ای با پس زمینه تقریباً ثابت و با توجه به ظاهر گوسی پارازیتها می‌توان تخمین زد که میانگین مقیاس خاکستری ناحیه B تقریباً نزدیک به میانگین مقیاس خاکستری تصویر بدون پارازیت است و این امر نشان می‌دهد که میانگین پارازیت صفر است. مقیاس خاکستری این ناحیه ثابت است بنا بر این تغییرپذیری ناحیه تعریف شده با b به علت اختلاف پارازیت است. (در صورت امکان روش دیگر برآورد میانگین و اختلاف پارازیت تصویربرداری از ناحیه خاکستری ثابت مورد نظر است). در تصویر ۵.۴.e پیشینه نما(histogram)ی یک مجموعه npix (در histroi) را می‌بینید که میانگین متغیرهای دلخواه گوسی آن ۱۴۷ است و اختلاف ۴۰۰ را دارند و با فرمانهای زیر حاصل شده‌اند:

```
>> X = imnoise2('gaussian', npix, 1, 147, 20);
>> figure, hist(X, 130)
>> axis([0 300 0 140])
```

در اینجا تعداد محفظه‌های hist طوری انتخاب شده که نتایج سازگار با تصویر ۵.۴.(c) باشد. پیشینه نما(histogram)ی این تصویر در تابع histroi با استفاده از imhist (رمز قبلی) که مقیاس بندی متفاوتی از hist دارد استفاده شد. برای تولید ایکس یک مجموعه متغیرهای دلخواه npix را انتخاب کردیم، طوری که تعداد نمونه‌ها در هر ۲ سابقه نما یکسان باشد. تشابه تصویرهای ۵.۴.(c) و d نمایانگر آن است که توزیع گوسی پارامترها نزدیک به برآورد v(1), v(2) باعث ایجاد پارازیت مشابه شده است.

۵.۳. بازگرداندن تصویر به حالت اول با حضور پارازیت و فیلترسازی فضایی

Restoration in Presence of Noise only-spatial filtering

روش منتخب برای کاهش پارازیت در این مورد فیلترسازی فضایی است و از شگردهایی شبیه به شگردهای بخش ۳.۴ و ۳.۵ استفاده می‌شود. در این بخش برای کاهش پارازیت چندین فیلتر فضایی را خلاصه و اجرا می‌کنیم. [Gonzalez & Woods 2002] سایر جزئیات خصوصیات این فیلترها را بحث کرده‌اند.

$$g(x, y) = f(x, y) + \eta(x, y)$$

۵.۳.۱. فیلترهای پارازیت فضایی(Spatial Noise Filters)

فیلترهای فضایی این بخش در جدول ۵.۲ فهرست بندی شده است. S_{xy} نمایانگر ناحیه فرعی m * n در داده‌های تصویر پارازیت‌دار g است. زیرنویسهای S نشان می‌دهند که تصویر فرعی در کانون مختصات (x, y) است و f(x, y) برآورد f نمایانگر واکنش فیلتر در آن مختصات است. فیلترهای خطی با استفاده از تابع imfilter بحث شده در بخش ۳.۴ تشریح شده‌اند. فیلترهای حداقل، حداکثر و میانگین غیرخطی و ترتیبی هستند. می‌توان فیلتر وسطی را مستقیماً با استفاده از تابع iptmedfilt2 محاسبه کرد. فیلترهای حداقل و حداکثر با استفاده از تابع مرتب کننده ordfilt2 بررسی شده در بخش ۳.۵.۲ اجرا می‌شوند.

تابع زیر موسوم به spfilt فیلترسازی را در دامنه فضایی با هر یک از فیلترهای فهرست بندی شده در جدول ۵.۲ انجام می‌دهد. به کاربرد تابع imlincomb در جدول ۲.۵ برای محاسبه ترکیب خطی داده‌های ورودی توجه کنید. ترکیب این تابع به شرح زیر است:

```
B = imlincomb(c1, A1, c2, A2, ..., ck, Ak)
```

که معادله زیر را اجرا می‌کند:

```
B = c1*A1 + c2*A2 + … + ck*Ak
```

در جایی که cها واقعی باشند کمیتهای عددی مضاعف و aها آرایه‌های عددی با همان نوع و اندازه هستند. ببینید چگونه تابع warning در تابع فرعی gmean فعال و غیرفعال می‌شود. در این حالت جلوی هشداری را می‌گیریم که اگر شناسه تابع log صفر شود در نرم افزار MATLAB صادر می‌شود. این تابع هشدار دهنده را می‌توان در هر برنامه‌ای استفاده رد. ترکیب اصلی آن به شرح زیر است:

```
warning('message')
```

عملکرد این تابع همچون تابع disp است. تنها فرقش آن است که می‌توان با فرمانهای فعال سازی و غیرفعال سازی هشدار آن را خاموش و روشن کرد.

```
function f = spfilt(g, type, m, n, parameter)
%SPFILT performs linear and nonlinear spatial filtering.
%     F = SPFILT(G, TYPE, M, N, PARAMETER) performs spatial filtering
%     of image G using a TYPE filter of size M-by-N Valid calls to
%      SPFILT are as follows:
%
%     F = SPFILT(G, 'amean', M, N)        Arithmetic mean filtering.
%     F = SPFILT(G, 'gmean', M, N)        Geometric mean filtering.
%     F = SPFILT(G, 'hmean', M, N)        Harmonic mean filtering.
%     F = SPFILT(G, 'chmean', M, N)       Contraharmonic mean
%                                         filtering of order Q. The
%                                         default is Q = 1.5.
%     F = SPFILT(G, 'median', M, N)       Median filtering.
%     F = SPFILT(G, 'max', M, N)          Max filtering.
%     F = SPFILT(G, 'min', M, N)          Min filtering.
%     F = SPFILT(G, 'midpint', M, N)            Midpoint filtering.
```

```
%     F = SPFILT(G, 'atrimmed', M, N)     Alpha-trimmed mean filtering.
%                                         parameter D must be a nonnegative
%                                         even integer; its default
%                                         value is D = 2.
%     The default values when only G and TYPE are input are M = N = 3,
%     Q = 1.5, and D = 2.

% Process inputs.
if nargin == 2
      m = 3; n = 3; Q = 1.5; d = 2;
elseif nargin == 5
      Q = parameter; d = parameter;
elseif nargin == 4
      Q = 1.5; d = 2;
else
      error('Wrong number of inputs.');
end

% Do the filtering.
Switch type
case 'amean'
      w = fspecial('average', [m n]);
      f = imfilter(g, w, 'replicate');
case 'gmean'
      f = gmean(g, m, n);
case 'hmean'
      f = harmean(g, m, n);
case 'chmean'
      f = charmean(g, m, n, Q);
case 'median'
      f = medfilt2(g, [m n], 'symmetric');
case 'max'
      f = ordfilt2(g, m*n, ones(m, n), 'symmetric');
case 'min'
      f = ordfilt2(g, 1, ones(m, n), 'symmetric');
case 'midpoint'
      f1 = ordfilt2(g, 1, ones(m, n), 'symmetric');
      f2 = ordfilt2(g, m*n, ones(m, n), 'symmetric');
      f = imlincomb(0.5, f1, 0.5, f2);
case 'atrimmed'
      if (d<0) | (d/2 ~= round(d/2))
            error('d must be a nonnegative, even integer');
      end
      f = alphatrim(g, m, n, d);
otherwise
      error('Unknown filter type');
end
%---------------------------------------------------------------------------%
function f = gmean(g, m, n)
%     Implement a geometric mean filter.
inclass = class(g);
g = im2double(g);
%     Disable log(0) warning.
warning off;
f = exp(imfilter(log(g), ones(m, n), 'replicate')).^(1/m/n);
warning on;
f = changeclass(inclass, f);
%---------------------------------------------------------------------------%
function f = harmean(g, m, n)
```

```
%     Implement a harmonic mean filter.
inclass = class(g);
g = im2double(g);
f = m * n ./ imfilter(1./(g + eps), ones(m, n), 'replicate');
f = changeclass(inclass, f);
%-------------------------------------------------------------------------------%
function f = charmean(g, m, n, q)
%     Implement a contrahamonic mean filter.
inclass = class(g);
g = im2double(g);
f = imfilter(g.^ (q+1), ones(m, n), 'replicate');
f = f ./ (imfilter(g.^q, ones(m, n), 'replicate') + eps);
f = changeclass(inclass, f);
%-------------------------------------------------------------------------------%
function f = alphatrim(g, m, n, d)
%     Implement a alpha-trimmed mean filter.
inclass = class(g);
g = im2double(g);
f = imfilter(g, ones(m, n), 'symmetric');
for k = 1:d/2
      f = imsubtract(f, ordfilt2(g, k, ones(m, n), 'symmetric'));
end
for k = (m*n - (d/2) + 1):m*n
      f = imsubtract(f, ordfilt2(g, k, ones(m, n), 'symmetric'));
end
f = f / (m*n – d);
f = changeclass(inclass, f)
```

TABLE 5.2 Spatial filters. The variables m and n denote respectively the number of rows and columns of the filter neighborhhood.

Filter Name	Equation	Comments
Arithmetic mean	$\hat{f}(x,y) = \frac{1}{mn}\sum_{(s,t)\in S_{xy}} g(s,t)$	Implemented using IPT functions `w = fspecial('average', [m, n])` and `f = imfilter(g, w)`.
Geometric mean	$\hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t)\right]^{\frac{1}{mn}}$	This nonlinear filter is implemented using function `gmean` (see custom function `spfilt` in this section).
Harmonic mean	$\hat{f}(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$	This nonlinear filter is implemented using function `harmean` (see custom function `spfilt` in this section).
Contraharmonic mean	$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$	This nonlinear filter is implemented using function `charmean` (see custom function `spfilt` in this section).
Median	$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{\text{median}}\{g(s,t)\}$	Implemented using IPT function `medfilt2`: `f = medfilt2(g, [m n])`.
Max	$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}}\{g(s,t)\}$	Implemented using IPT function `ordfilt2`: `f = ordfilt2(g, m*n, ones(m, n))`.
Min	$\hat{f}(x,y) = \min_{(s,t)\in S_{xy}}\{g(s,t)\}$	Implemented using IPT function `ordfilt2`: `f = ordfilt2(g, 1, ones(m, n))`.
Midpoint	$\hat{f}(x,y) = \frac{1}{2}\left[\max_{(s,t)\in S_{xy}}\{g(s,t)\} + \min_{(s,t)\in S_{xy}}\{g(s,t)\}\right]$	Implemented as 0.5 times the sum of the max and min filtering operations.
Alpha-trimmed mean	$\hat{f}(x,y) = \frac{1}{mn-d}\sum_{(s,t)\in S_{xy}} g_r(s,t)$	The $d/2$ lowest and $d/2$ highest intensity levels of $g(s,t)$ in S_{xy} are deleted, $g_r(s,t)$ denotes the remaining $mn - d$ pixels in the neighborhood. Implemented using function `alphatrim` (see custom function `spfilt` in this section).

جدول ۵.۲. فیلترهای فضایی: متغیرهای m, n به ترتیب نمایانگر تعداد ردیف و ستونهای ناحیه همجوار فیلتر هستند.

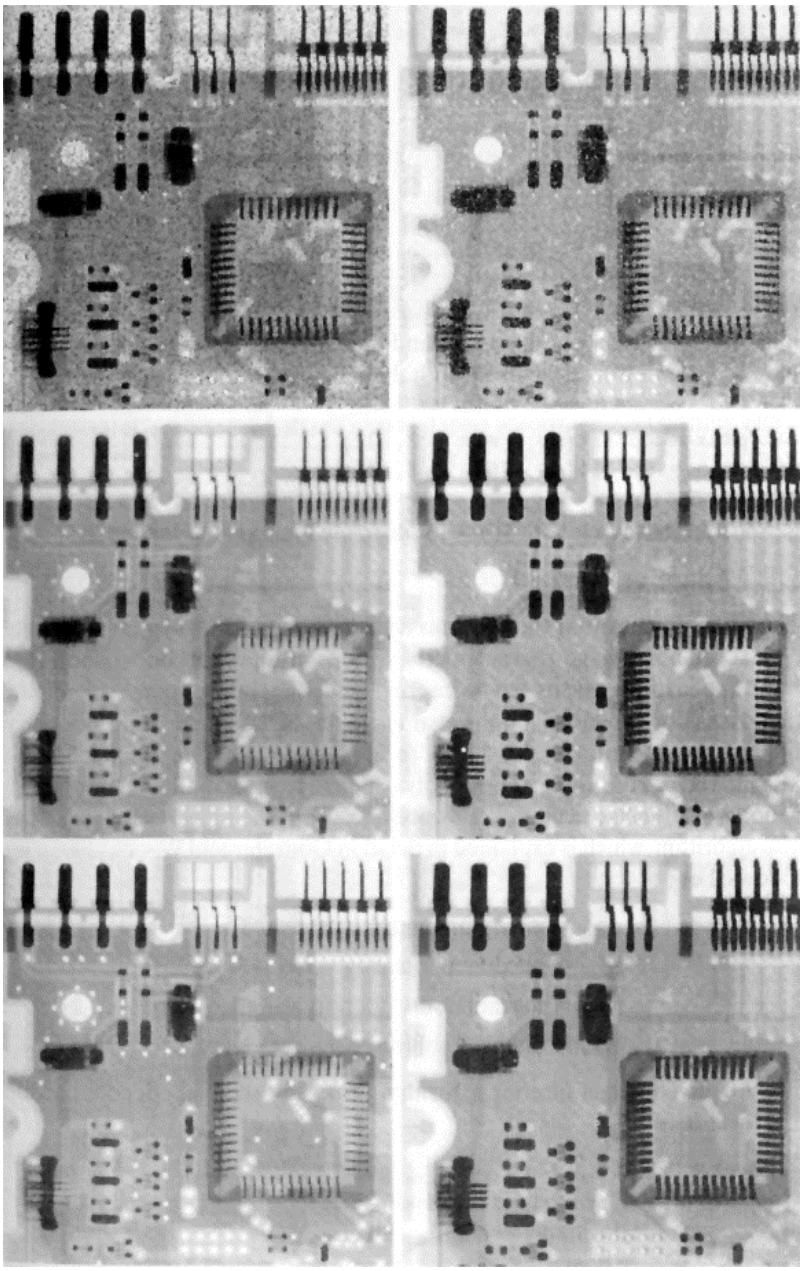
نام فیلتر	توضیحات
میانگین ریاضی	اجرا شده با استفاده از تابع‌های IPT
میانگین هندسی	این فیلتر غیرخطی با استفاده از تابع gmean اجرا می‌شود (به تابع سفارشی spfilt در این بخش مراجعه کنید)
میانگین هماهنگ	این فیلتر غیرخطی با استفاده از تابع harmean اجرا می‌شود (به تابع سفارشی spfilt در این بخش مراجعه کنید)
میانگین ناهماهنگ	این فیلتر غیرخطی با استفاده از تابع charmean اجرا می‌شود (به تابع سفارشی spfilt در این بخش مراجعه کنید)
میانگین	اجرا شده با استفاده از تابع IPT زیر: F=medfilt2(g,[m n])
حداکثر	اجرا شده با استفاده از تابع IPT زیر: F=ordfilt2(g,m*n,ones[m n])
حداقل	اجرا شده با استفاده از تابع IPT زیر: F=ordfilt2(g,1,ones[m n])
نقطه میانی	اجرا شده به صورت نیم برابر مجموع حداقل و حداکثر عملیات فیلترسازی
میانگین بریده شده آلفا	حداقل و حداکثر شدت d/2 در gstst و sxy حذف شده است. و مابقی عنصرهای تصویری mn-d در همجوار آن عرضه شده است. اجرا شده با استفاده از تابع آلفا تریم (به تابع سفارشی spfilt در این بخش مراجعه کنید)

مثال ۵.۵ :

تصویر ۵.۵.(a) یک تصویر uint8 تحریف شده با پارازیت افشان (Pepper noise) با احتمال ۰.۱ است. این تصویر با فرمانهای زیر ایجاد شد. f نمایانگر تصویر اصلی یعنی تصویر ۳.۱۸.(a) است.

```
>> [M, N] = size(f);
>> R = imnoise2('salt & pepper', M, N, 0.1, 0);
>> c = find(R == 0);
>> gp = f;
>> gp(c) = 0;
```

تصویر ۵.۵.(b) که با پارازیت افشان (Pepper noise) تحریف شده است با استفاده از عبارتهای زیر تولید شد.



تصویر ۵.۵. (a)) تصویر تحریف شده با پارازیت افشان (Pepper noise) با احتمال ۰.۱ (b) تصویر تحریف شده با پارازیت افشان (Pepper noise) با همان احتمال (c) نتیجه فیلترسازی ۳*۳ فیلتر ناهماهنگ با ترتیب ۱.۵ (d) نتیجه فیلترسازی ۳*۳ فیلتر حداکثر (e) نتیجه فیلترسازی ۳*۳ فیلتر حداقل

```
>> R = imnoise2('salt & pepper', M, N, 0, 0.1);
>> c = find(R == 1);
>> gs = f;
>> gs(c) = 255;
```

یک روش خوب برای فیلترسازی پارازیت افشان (Pepper noise) استفاده از فیلتر ناهماهنگ با مقدار مثبت Q است. تصویر ۵.۵.(c) با استفاده از عبارت زیر ایجاد شد:

```
>> fp = spfilt(gp, 'chmean', 3, 3, 1.5);
```

پارازیت افشان (Pepper noise) را می‌توان با استفاده از فیلتر ناهماهنگ با مقدار منفی Q فیلتر کرد.

```
>> fs = spfilt(gs, 'chmean', 3, 3, -1.5);
```

نتیجه در تصویر ۵.۵. d نشان داده شده است. با فیلترهای حداقل و حداکثر می‌توان نتایج مشابه به دست آورد. مثلاً تصاویر ۵.۵.ایی و f به ترتیب از تصاویر ۵.۵.(a) و (b) با فرمانهای زیر به دست آمد.

```
>> fpmax = spfilt(gp, 'max', 3, 3);
>> fpmin = spfilt(gs, 'min', 3, 3);
```

سایر راه حل‌ها با استفاده از spfilt به شکلی مشابه اجرا شدند.

۵.۳.۲. فیلترهای فضایی سازگار(Adaptive Spatial Filter)

فیلترهای بحث شده در بخش قبل بدون در نظر گرفتن تغییرات خصوصیات تصاویر به آنها اعمال می‌شوند. در برخی برنامه‌ها می‌توان برای بهبود نتایج از فیلترهایی استفاده کرد که رفتار آنها با توجه به خصوصیات ناحیه فیلتر شده تغییر می‌کند. برای نشان دادن نحوه اجرای فیلترهای فضایی سازگار در نرم افزار MATLAB یک فیلتر میانی سازگار را در این بخش در نظر می‌گیریم. S_{xy} نمایانگر تصویر فرعی در کانون (x, y) است الگوریتمی که Gonzalez & Woods[2002] آن را شرح داده‌اند به شکل زیر است:

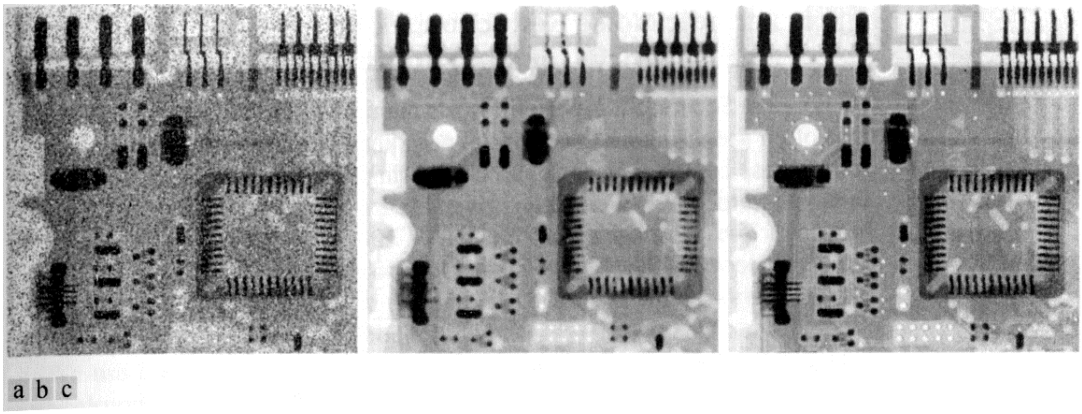
$z_{\min}$ = minimum intensity value in S_{xy}
$z_{\max}$ = maximum intensity value in S_{xy}
z_{med} = median of the intensity value in S_{xy}
z_{xy} = intensity value at coordinates (x, y)

الگوریتم فیلترسازی میانی سازگار به ۲ شکل کار می‌کنند که آنها را سطوح A,B می‌نامیم.

Level A: If $z_{\min} < z_{med} < z_{\max}$, go to level B
Else increase the window size
If window size $\leq z_{\max}$, repeat level A
Else output z_{med}

Level B: If $z_{\min} < z_{xy} < z_{\max}$, output z_{xy}
Else output z_{med}

در اینجا S_{max} نمایانگر حداکثر اندازه مجاز فیلتر سازگار است. گزینه دیگر در آخرین مرحله A آن است که به جای میانگین Z_{xy} را در داده‌های خروجی قرار دهیم. در این صورت تیرگی تصویر کمتر می‌شود ولی پارازیت افشان (Pepper noise) تعبیه شده در پس زمینه ثابت و هم اندازه با آن را نمی‌تواند کشف کند.



تصویر ۵.۶: (a)) تصویر تحریف شده با پارازیت افشان (Pepper noise) با چگالی ۰.۲۵ (b) نتیجه به دست آمده با استفاده از فیلتر حد واسط به اندازه ۷*۷ (c) نتیجه به دست آمده با استفاده از فیلتر سازگار با Smax=7

یک تابع M که این الگوریتم را اجرا می‌کند و ما آن را adpmedian می‌نامیم:

```
f = adpmedian(g, Smax)
```

در اینجا g تصویر فیلتر شده است و همان طور که در بالا تعریف کردیمS_{max} حداکثر اندازه مجاز فیلتر سازگار است:

در تصویر ۵.۶.(a) تصویر مدار f که با پارازیت افشان (Pepper noise) تحریف شده و با فرمان زیر ایجاد شده است دیده می‌شود:

نتیجه به دست آمده با استفاده از این فرمان (بخش ۳.۵.۲ در خصوص استفاده از medfilt2) نشان داده شده است

```
>> g = imnoise(f, 'salt & pepper', .25);
>> f1 = medfilt2(g, [7 7], 'symmetric');
```

این تصویر بدون پارازیت است ولی تیره و تحریف شده است. (به رابطهای بالای کانون تصویر مراجعه کنید). از طرف دیگر، فرمان زیر

```
>> f2 = adpmedian(g, 7);
```

نتیجه مندرج در تصویر ۵.۶.(c) را ارائه داد که عاری از پارازیت است ولی تیرگی و تحریف آن کمتر از تصویر ۵.۶.(b) است.

5.4 .کاهش متناوب پارازیت با فیلترسازی دامنه فرکانس

Periodic Noise Reduction by Frequency Domain Filtering

همان طور که در بخش ۵.۲.۳ گفتیم پارازیت متناوب به شکل ضربه‌هایی دیده می‌شود که در طیف فوریر قابل رویت هستند. روش اصلی برای فیلترسازی این مولفه‌ها فیلترسازی شکافدار است. تابع انتقال فیلتر شکاف‌دار باتروورث از نوع n در فرمول زیر دیده می‌شود:

$$H(u,v)=\frac{1}{1+[\frac{D_0^2}{D_1(u,v)D_2(u,v)}]^n}$$

$$D_1(u,v)=[(u-M/2-u_0)^2+(u-N/2-v_0)^2]^{1/2}$$

$$D_2(u,v)=[(u-M/2-u_0)^2+(u-N/2-v_0)^2]^{1/2}$$

در اینجا $(-u_0,-v_0)$, (u_0,v_0) و تقارن آن موضع شکافها هستند و D_0 عامل سنجش شعاع است. توجه داشته باشید که این فیلتر با توجه به کانون مستطیل فرکانس مشخص شده است بنا بر این قبل از استفاده باید با تابع fftshift پردازش شود که در بخشهای ۴.۲ و ۴.۳ تشریح شده است.

نوشتن تابع M برای فیلترسازی شکاف‌دار تابع همان اصول مورد استفاده در بخش ۴.۵ است. بهتر است تابع طوری نوشته شود که چندین شکاف را بتوان در داده‌های ورودی قرار داد همانند روش مورد استفاده در بخش ۵.۲.۳ برای ایجاد الگوهای پارازیت منحنی چندگانه. پس از به دست آوردن H فیلترسازی با استفاده از تابع dftfilt که در بخش ۴.۳.۳ تشریح شد ادامه می‌یابد.

۵.۵. الگوبرداری از تابع کاهش رنگ(Modeling Degradation Function)

وقتی تجهیزاتی شبیه به تجهیزات تولید تصویر با رنگ کاهش یافته موجود باشند، می‌توان ماهیت روند کاهش رنگ را با تغییر تنظیم تجهیزات تجربه کرد. ولی در دسترس بودن این تجهیزات برای حل مسائل مربوط به بازگرداندن تصویر به حالت اول یک اصل نیست و یک

روش متداول آزمایش کردن با تولید PSF است و سپس بررسی نتایج با الگوریتم‌های گوناگون برای بازگرداندن تصویر به حالت اول است. روش دیگر الگوبرداری ریاضی از PSF است. این روش خارج از مبحث این قسمت است

برای بررسی این موضوع به مطالب Gonzalez & Woods[2002] مراجعه کنید. وقتی هیچ اطلاعاتی در خصوص PSF نداشته باشیم می‌توان برای استنتاج PSF تلفیق‌زدایی کورکورانه انجام داد. این روش در بخش ۵.۱۰ تشریح شده است. نکته اصلی مابقی این بخش شگردهای گوناگون الگوبرداری PSF با استفاده از تابعهای imfilter, fspecial است که به ترتیب در بخشهای ۳.۴ و ۳.۵ معرفی شدند. برخی تابعهای گوناگون تولید پارازیت در این بخش تشریح شده است.

یکی از مسائل مهم کاهش رنگ که در بازگرداندن تصویر به حالت اول دیده می‌شود تیرگی یا تاری تصویر است. این نوع تیرگی که همزمان در حسگر و صحنه‌ها دیده می‌شود با فیلترهای پایین گذر دامنه فرکانس یا فیلترهای فضایی قابل الگوبرداری است.

الگوی مهم دیگر در کاهش رنگ تیرگی آن به دلیل حرکت خطی ثابت بین حسگر و صحنه حین تصویربرداری است. این تیرگی صحنه را می‌توان با استفاده از تابع fspecial , IPT الگوبرداری کرد.

```
PSF = fspecial('motion', len, theta)
```

این نوع فراخوانی fspecial باعث تولید PSFی می‌شود که می‌تواند با عناصر تصویری len تاثیری شبیه به تاثیر حرکت خطی دوربین ایجاد کند. پارامتر thete بر حسب درجه است و با توجه به محور افقی مثبت در خلاف جهت عقربه‌های ساعت سنجیده می‌شود. مقدار پیش فرض len و thete به ترتیب ۹ و ۰ است که مطابق با حرکت ۹ عنصر تصویری در جهت افقی است.

از تابع imfilter برای ایجاد تصویری با رنگهای کاهش یافته استفاده می‌کنیم که PSF آن یا مشخص است و یا با استفاده از روش قبل محاسبه می‌شود.

```
>> g = imfilter(f, PSF, 'circular');
```

در اینجا جدول ۳.۲ برای کاهش تاثیرات مرزی به کار برده می‌شود. سپس با افزودن پارازیت الگوی تصویر با رنگهای کاهش یافته را تکمیل می‌کنیم.

```
>> g = g + noise;
```

در اینجا noise یک تصویر پارازیت‌دار به همان اندازه g است که با استفاده از یکی از روشهای بخش ۵.۲ تولید شده است.

وقتی میزان مناسب بودن روشهای گوناگون این بخش و بخش بعدی را با هم مقایسه می‌کنیم می‌بینیم که بهتر است از یک تصویر یا الگوی آزمایشی استفاده کنیم تا مقایسه‌ها با معنی باشد. الگوی آزمایشی تولید شده با تابع چکربورد برای این منظور بسیار مناسب است زیرا می‌توان اندازه آن را بدون تاثیرگذاری بر خصوصیات اصلی کاهش داد. ترکیب آن به شرح زیر است:

```
C = checkerboard(NP, M, N)
```

در اینجا NP تعداد عناصر تصویری در هر طرف مربع است. M تعداد ردیفها است. N تعداد ستونها است. اگر N حذف شود، مقدار پیش فرض آن M می‌شود. اگر N, M هر دو حذف شوند، یک مربع شطرنجی با ۸ ضلع در کنار ایجاد می‌شود. اگر NP حذف شود، مقدار پیش فرض آن ۱۰ عنصر تصویری می‌شود. مربع‌های روشن در نیمه چپ صفحه شطرنجی سفید هستند. مربع‌های روشن در نیمه راست صفحه شطرنجی خاکستری هستند. برای ایجاد صفحه شطرنجی که همه مربع‌های آن سفید باشند از فرمان زیر استفاده می‌کنیم:

```
>> K = im2double(checkerboard(NP, M, N)) > 0.5;
```

تصاویر ایجاد شده با تابع شطرنجی از نوع مضاعف با مقادیری در حیطه [0,1] هستند.

از آنجائی که برخی الگوریتم‌های بازگرداندن تصویر به حالت اول کند عمل می‌کنند یک روش خوب آزمایش روی تصاویر کوچک برای کاهش زمان محاسبه و بهبود کنش و واکنش است. در این حالت بهتر است که با تکثیر عنصرهای تصویری بتوان روی تصویر متمرکز شد.

استفاده از عملگر > باعث تولید نتیجه منطقی می‌شود: im2double برای ایجاد تصویر مضاعف به کار برده می‌شود که با فرمت خروجی تابع شطرنجی سازگار است. تابع زیر این کار را انجام می‌دهد:

```
B = pixeldup(A, m, n)
```

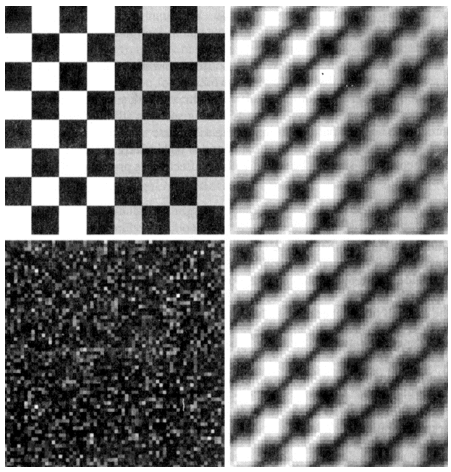
این تابع هر عنصر a را به تعداد m دفعه در جهت عمودی و n دفعه در جهت افقی تکثیر می‌کند. اگر n حذف شود مقدار پیش فرض آن m می‌شود.

مثال ۵.۷: الگوبرداری از تصویر پارازیت‌دار تیره :تصویر شطرنجی ایجاد شده با فرمان زیر در تصویر ۵.۷.(a) دیده می‌شود.

```
>> f = checkerboard(8);
```

تصویر ۵.۷.(b) با رنگهای کاهش یافته با استفاده از فرمان زیر ایجاد شده است.

```
>> PSF = fspecial('motion', 7, 45)
>> g = imfilter(f, PSF, 'circular');
```



تصویر ۵.۷: (a)) تصویر اصلی (b) تصویر تیره شده (c) تصویر پارازیت‌دار (d) مجموع (b) و پ

توجه داشته باشید که PSF فقط یک فیلتر فضایی است.

```
>> PSF
PSF =
0          0          0          0          0          0.0145     0
0          0          0          0          0.0376     0.1283     0.0145
0          0          0          0.0376     0.1283     0.0376     0
0          0          0.0376     0.1283     0.0376     0          0
0          0.0376     0.1283     0.0376     0          0          0
0.0145     0.1283     0.0376     0          0          0          0
0          0.0145     0          0          0          0          0
```

الگوی پارازیت‌دار تصویر ۵.۷(c) با استفاده از فرمان زیر ایجاد شده است.

```
>> noise = imnoise(zeros(size(f)), 'gaussian', 0, 0.001);
```

معمولاً پارازیت را مستقیماً با استفاده از imnoise(gb,Gaussian,0,0.001) گوسی ۰.۰۰۱ اضافه می‌کنیم. ولی تصویر پارازیت‌دار بعداً در این فصل مورد نیاز می‌شود بنا بر این آن را در اینجا جداگانه محاسبه کرده‌ایم.

تصویر پارازیت‌دار تیره ۵.۷. d با استفاده از فرمان زیر ایجاد شده است.

```
>> g = gb + noise;
```

پارازیت به راحتی در این تصویر دیده نمی‌شود زیرا حداکثر مقدار آن در ردیف ۰.۱۵ است در صورتی که حداکثر مقدار تصویر ۱ است. همان طور که در بخشهای ۵.۷ و ۵.۸ دیده می‌شود وقتی می‌خواهیم g را به حالت اصلی بازگردانیم سطح پارازیت مهم نیست. کلیه تصاویر عکس ۵.۷ به اندازه ۵۱۲*۵۱۲ بزرگ شدند و با استفاده از فرمان زیر نمایش داده شدند:

```
>> imshow(pixeldup(f, 8), [ ])
```

عکس تصویر ۵.۷. d در مثالهای ۵.۸ و ۵.۹ به حالت اصلی بازگردانده شده است.

۵.۶. فیلترسازی معکوس مستقیم(Direct Inverse filtering)

ساده‌ترین روش برای بازگرداندن تصویر مستهلک شده تخمین رابطه زیر است:

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$$

سپس با توجه به تبدیل معکوس فوریر f(u, v) تصویر مربوطه برآورد می‌شود توجه داشته باشید که g(u, v) تبدیل فوریر در تصویر مستهلک شده است. این روش را فیلترسازی معکوس می‌نامیم. از الگوی بخش ۵.۱ می‌توان برآورد زیر را انجام داد:

$$\hat{F}(u,v) = F(u,v) + \frac{N(u,v)}{H(u,v)}$$

در این عبارت به ظهر ساده می‌بینیم که اگر h(u, v) را دقیقاً بدانیم f(u, v) و تصویر اصلی مستهلک نشده را نمی‌توان بازیافت زیرا مولفه پارازیت یک تابع دلخواه است که تبدیل فوریر آن شناخته شده نیست. مسئله تابع h(u, v) آن است که صفرهای متعدد دارد. حتی اگر از عبارت n(u, v) صرف نظر کنیم، تقسیم آن بر h(u, v) بر این برآوردها می‌چربد.

روش متداول حین فیلترسازی معکوس تشکیل نسبت f(u, v) = g(u, v) / h(u, v) است. سپس معکوس‌سازی دامنه فرکانس برای به دست آوردن فرکانسهای نزدیک مبداء. صفرهای h(u, v) کمتر احتمال دارد که نزدیک مبداء روی دهند زیرا بزرگی تبدیل در این موضع در حداکثر است. این مضمون گونه‌های متفاوت دارد که در مقادیر (u, v) که H آن صفر یا نزدیک به صفر باشد عملیات خاص اجرام می‌شود. این روش را گاهی فیلترسازی کاذب معکوس می‌نامیم. به طور کلی روشهای مبتنی بر فیلترسازی معکوس از این نوع به ندرت کاربردی هستند که در مثال ۵.۸ بخش بعد دیده می‌شود.

۵.۷. فیلترسازی Weiner

این روش فیلترسازی اولین مرتبه توسط Weiner در ۱۹۴۲ پیشنهاد شد و یکی از اولین و شناخته شده ترین روشهای بازگرداندن خطی تصویر به حالت اول است. فیلتر Weiner f را تخمین می‌زند و تابع خطای آماری را کمینه می‌کند

$$e^2 = E\{(f - \hat{f})^2\}$$

در اینجا E عملگر مورد انتظار مقادیر است و f تصویر مستهلک نشده است. راه حل این عبارت در دامنه فرکانس به شرح زیر است:

$$\hat{F}(u,v) = [\frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}]G(u,v)$$

$H(u,v) = the\ \text{deg}radation\ function$

$|H(u,v)|^2 = H^*(u,v)H(u,v)$

$H^*(u,v) = the\ complex\ conjugate\ of\ H(u,v)$

$S_\eta(u,v) = |N(u,v)|^2 = the\ power\ spectrum\ of\ the\ noise$

$S_\eta(u,v) = |F(u,v)|^2 = the\ power\ spectrum\ of\ the\ un\text{deg}raded\ image$

نسبت $s_\eta(u, v)/ s_f (u, v)$ نسبت پارازیت به علائم نامیده می‌شود. اگر طیف پارازیت برای همه مقادیر u, v صفر باشد این نسبت صفر می‌شود و فیلتر Weiner تا حد فیلتر معکوس‌سازی مورد بحث در بخش قبل کاهش می‌یابد.

دو مقدار مورد بحث در این زمینه میانگین قدرت پارازیت و تصویر است.

$$f_A = \frac{1}{MN}\sum_u \sum_v S_f(u,v)$$

در اینجا M,N به ترتیب نمایانگر اندازه عمودی و افقی تصویر و آرایه‌های پارازیت هستند. این مقادیر ثابتهای مقیاس هستند و نسبت آنها نیز که یک مقیاس است برای ایجاد یک ارایه ثابت به جای تابع به کار برده می‌شود.

$$R = \frac{\eta A}{f_A}$$

حتی اگر نسبت واقعی مشخص نباشد می‌توان با چند آزمایش ساده مقادیر ثابت را تغییر داد و نتایج بازگرداندن تصویر به حالت اول را مشاهده کرد. در این روش فرض می‌شود تابع‌ها ثابت هستند. جایگزین کردن $s_\eta(u, v)/ s_f(u, v)$ با آرایه ثابت در معادله فیلتر قبل باعث ایجاد فیلتر پارامتری Weiner می‌شود. همان طور که در مثال ۵.۸ می‌بینید استفاده از آرایه ثابت امتیازهای مهمی نسبت به فیلترسازی معکوس مستقیم دارد.

فیلترسازی Weiner در IPT با استفاده از تابع deconvwnr انجام می‌شود که سه ترکیب احتمالی دارد. در تمام این موارد g نمایانگر تصویر مستهلک شده و fr نمایانگر بازگرداندن تصویر به حالت اول است. در شکل اول این ترکیب

```
fr = deconvwnr(g, PSF)
```

فرض می‌شود که نسبت پارازیت به علائم صفر است. بنا بر این، این نوع فیلتر Weiner فیلتر معکوس قید شده در بخش ۵.۶ است. در ترکیب زیر

```
fr = deconvwnr(g, PSF, NSPR)
```

فرض می‌شود که نسبت پارازیت به علائم به صورت ثابت یا در ارایه شناخته شده است. تابع یکی از این دو را قبول می‌کند. این ترکیبی است که برای اجرای فیلتر پارامتری Weiner به کار برده می‌شود. در این صورت، nspr داده‌های مقیاس ورودی را دارد. در ترکیب زیر

```
fr = deconvwnr(g, PSF, NACORR, FACORR)
```

فرض می‌شود که تابعهای همبستگی NACORR,FACORR در پارازیت و تصویر مستهلک نشده شناخته شده هستند. توجه داشته باشید که این نوع deconvwr به جای استفاده از طیف این تابعها از همبستگی η, f بهره می‌برد. رابطه زیر با توجه به قضیه همبستگی مصداق دارد:

$$|F(u,v)|^2 = \Im[f(x,y) \circ f(x,y)]$$

در اینجا o نمایانگر عملیات همبستگی است و 3 نمایانگر تبدیل فوریر است.

در این عبارت می‌بینیم که می‌توان تابع همبستگی را برای استفاده در deconvwr با محاسبه تبدیل معکوس فوریر در این طیف به دست آورد. همین موضوع در خصوص همبستگی خودکار پارازیت نیز صدق می‌کند.

اگر تصویر تعمیر شده حلقه‌های تبدیل متمایز فوریر را که در الگوریتم استفاده می‌شوند نشان دهد می‌توان قبل از فراخوانی deconvwnr از تابع edgetaper استفاده کرد.

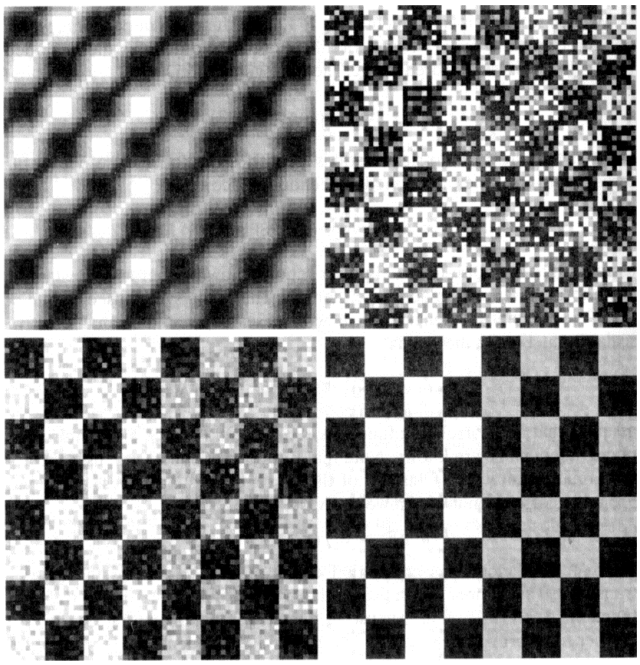
```
J = edgetaper(I, PSF)
```

در این تابع لبه‌های تصویر ورودی با استفاده از تابع گسترش نقاط PSF تیره می‌شود. تصویر موجود در داده‌های خروجی j از ادغام داده‌های I و تصویر تیره شده به دست آمده است. آرایه سنجشگر که با تابع همبستگی خودکار PSF تعیین می‌شود باعث می‌شود j در کانون خود معادل I شود. و در لبه‌ها معادل تصویرهای تیره شده شود.

مثال ۵.۸: استفاده از تابع deconvwr برای تعمیر تصویر پارازیتد r تار شده

تصویر ۵.۸.(a) همان تصویر ۵.۷. d و ۵.۸.(b) است که با استفاده از فرمان زیر به دست آورده شد.

```
>> fr1 = deconvwnr(g, PSF);
```



تصویر ۵.۸. (a)) تصویر پارازیت‌دار تیره (b) نتیجه فیلترسازی معکوس (c) نتیجه فیلترسازی Weiner با استفاده از نسبت ثابت (d) نتیجه فیلترسازی Weiner با استفاده از تابعهای همبستگی خودکار

g تصویر تحریف شده است PSF تابع گسترش نقطه است که در مثال ۵.۷ محاسبه شد. همان طور که قبلاً در این بخش گفتیم fr1 نتیجه فیلترسازی معکوس است و این نتایج تحت تاثیر پارازیت قرار گرفته است. (همان طور که در مثال ۵.۷ گفتیم کلیه تصویرهای نمایش داده شده با pixeldup پردازش شدند تا اندازه آنها به ۵۱۲*۵۱۲ پیکسل برسد.

نسبت r که قبلاً در این بخش تشریح شده است با استفاده از تصویرهای اصلی و پارازیت‌دار مثال ۵.۷ به دست آمد.

```
>> Sn = abs(fft2(noise)).^2;              % noise power spectrum
>> nA = sum(Sn(:))/prod(size(noise));     % noise average power
>> Sf = abs(fft2(f)).^2;                  % image power spectrum
>> fA = sum(Sf(:))/prod(size(f));         % image average power
>> R = nA/fA;
```

برای تعمیر این تصویر با استفاده از نسبت زیر عبارت زیر را می‌نویسیم:

```
>> fr2 = deconvwnr(g, PSF, R);
```

همان طور که در تصویر ۵.۸.(c) دیده می‌شود این روش نسبت به فیلترسازی معکوس مستقیم امتیاز مهمی دارد.

در آخر برای بازگرداندن تصویر به حالت اصلی از تابع همبستگی خودکار استفاده می‌کنیم. (به کاربرد fftshift برای تمرکزیابی توجه کنید)

```
>> NCORR = fftshift(real(ifft2(Sn)));
>> ICORR = fftshift(real(ifft2(Sf)));
>> fr3 = deconvwnr(g, PSF, NCORR, ICORR);
```

در تصویر ۵.۸. d دیده می‌شود که نتیجه نزدیک به حالت اصلی است ولی هنوز مقداری پارازیت دارد. از آنجائی که تصویر اصلی و تابع‌های پارازیت مشخص هستند توانستیم پارامترهای صحیح را تخمین بزنیم. تصویر ۵.۸. d بهترین تصویری است که با تلفیق‌زدایی Weiner تحقق یافتنی است. وقتی یک یا چند کمیت فوق مشخص نباشند مشکل انتخاب هوشمند تابعهای مورد استفاده در آزمایش تا زمان به دست آوردن نتیجه قابل قبول است.

۵.۸. فیلترسازی (منظم) با استفاده از کوچکترین مربع محدود

Constrained Least Squares (Regularized) Filtering

یک روش شناخته شده دیگر برای بازگرداندن تصویر به حالت اصلی روش فیلترسازی (منظم) با استفاده از کوچکترین مربع محدود است که در اسناد IPT فیلترسازی منظم نامیده می‌شود. تلفیق متمایز ۲ بعدی به شکل زیر تعریف می‌شود:

$$h(x,y) * f(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) h(x-m, y-n)$$

با استفاده از این معادله می‌توان الگوی کاهش خطی رنگ را که در بخش ۵.۱ تشریح شده است بیان کرد.

$$g = Hf + \eta$$

مثلاً فرض کنید $g(x, y)$ به اندازه $M*N$ است. در این صورت، می‌توان اولین عنصرهای n بردار g را با استفاده از عناصر تصویری اولین ردیف $g(x, y)$ درست کرد، عنصرهای بعدی n از ردیف دوم درست می‌شود و الی آخر. بردار حاصل از آن ابعاد $mn*1$ را خواهد داشت. اینها هم ابعاد f, η هستند و این بردارها هم به همین شکل ایجاد می‌شوند. بنا بر این ماتریس H ابعاد $mn*mn$ را دارد. عناصر آن توسط عناصر معادله تلفیق قبلی ارائه شده است.

حالا می‌توانیم مسئله بازگرداندن تصویر به حالت اصلی را به یک سری عملیات ماتریسی ساده تجزیه کنیم. متاسفانه مشکل این نیست. فرض کنید با تصویرهای متوسط داریم کار می‌کنیم. مثلاً $m = n = 512$ در این صورت، بردارهای معادله ماتریسی قبل به ابعاد ۲۶۲، ۱*۱۴۴ هستند و ابعاد ماتریس H ۲۶۲، ۱۴۴*۲۶۲ است. مدیریت بردارها و ماتریسهایی به این اندازه کار ساده‌ای نیست. نکته‌ای که مسئله را پیچیده‌تر می‌کند آن است که معکوس H به دلیل وجود صفر در تابع انتقال همیشه وجود ندارد (بخش ۵.۶). ولی فرمول‌بندی مسئله تعمیر تصویر در ماتریس باعث تسهیل اشتقاق شگردهای مسئله بازگرداندن تصویر به حالت اصلی می‌شود.

گرچه از روش کوچکترین مربعات محدود که در اینجا ارائه شده است استفاده نمی‌کنیم، ولی مسئله اصلی در اینجا حساسیت معکوس H قید شده در پاراگر f قبل است. یک راه حل این مسئله آن است که این کار بر اساس معیاری هموار مثلاً دومین مشتق تصویر (مثلاً در لاپلاس) صورت گیرد. برای این که مسئله بازگرداندن تصویر به حالت اصلی معنی‌دار باشد باید آن را با پارامترهای این مسئله محدود کنیم. بنا بر این باید حداقل معیار تابع cرا طبق تعریف زیر به دست آوریم.

?

که ممکن است به شکل زیر محدود شود:

?

در اینجا $||w||^2=w^Tw$ هنجار بردار اقلیدسی است. F برآورد تصویر مستهلک نشده است. و عملگر لاپلاس در بخش ۳.۵.۱ تعریف شد. راه حل دامنه فرکانس برای بهینه‌سازی این مسئله در عبارت زیر ارائه شده است.

$$\hat{F}(u,v)=[\frac{|H(u,v)|}{|H(u,v)|^2+\gamma|P(u,v)|^2}]G(u,v)$$

γ پارامتری است که باید طوری سازگار شود که محدودیت درست ایجاد شود (اگر γ صفر باشد راه حل فیلتر معکوس است) و $p(u, v)$ تبدیل فوریر تابع است.

$$p(x,y)=\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}$$

این تابع عملگر لاپلاس است که در بخش ۳.۵.۱ معرفی شد. تنها مجهولهای فرمول قبل $\|\eta\|^2$,γ هستند. اگر $\|\eta\|^2$ که متناسب با قدرت پارازیت است کمیت عددی مشخص باشد، γ را می‌توان با تکرار پیدا کرد.

فیلترسازی کوچکترین مربعات محدود توسط تابع deconvreg در IPT اجرا می‌شود که ترکیب زیر را دارد:

```
fr = deconvreg(g, PSF, NOISEPOWER, RANGE)
```

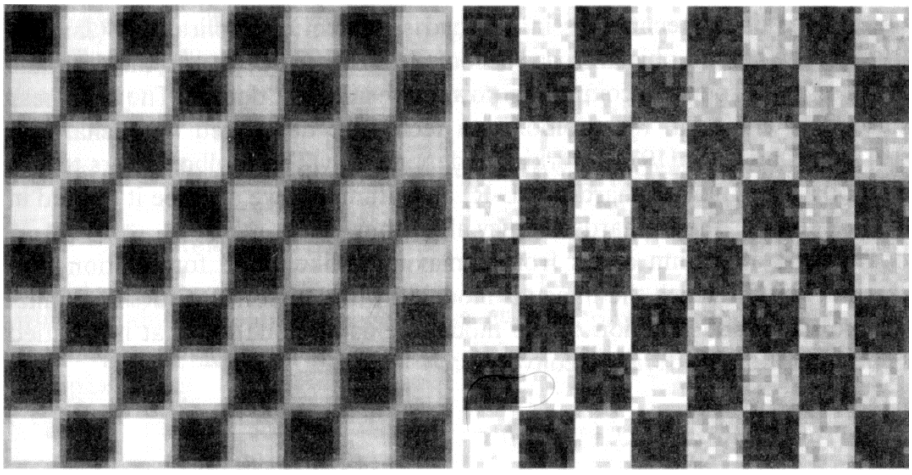
در اینجا g تصویر تحریف شده است. Fr تصویر تعمیر شده است، NOISEPOWER متناسب با $\|\eta\|^2$ است و RANGE حیطه مقادیری است که این الگوریتم برای یافتن راه حل γ آنها را جستجو می‌کند. مقدار پیش فرض ([1e-10,1e10]) $[10^{-9},10^{9}]$ در نشانه‌گذاریهای نرم افزار MATLAB است. اگر ۲ پارامتر آخر از این شناسه حذف شوند، deconvreg راه حل فیلتر معکوس را تولید می‌کند. برآورد مقدماتی خوب برای NOISEPOWER, $MN[\delta_\eta^2+m_\eta^2]$ است. در اینجا m , n ابعاد تصویر هستند و پارامترهای داخل کروشه اختلاف پارازیت و میانگین مربعات پارازیت هستند.این تخمین مقدماتی است و همان طور که در مثال بعد می‌بینید مقدار نهایی ممکن است متفاوت باشد.

مثال ۵.۹: استفاده از تابع deconvreg برای بازگرداندن تصویر پارازیت‌دار تار به حالت اصلی

حالا با استفاده از deconvreg تصویر ۵.۷. d را به حالت اصلی بازمی‌گردانیم. اندازه این تصویر ۶۴*۶۴ است و از مثال ۵.۷ می‌فهمیم که اختلاف پارازیت ۰.۰۰۱ است و به میانگین پارازیت پی می‌بریم.

بنا بر این تخمین مقدماتی ما از NOISEPOWER , $4 \sim [0.001 -0](64)^2$است. نتایج استفاده از فرمان زیر در تصویر ۵.۹.(a) نشان داده شده است.

```
>> fr = deconvreg(g, PSF, 4);
```



تصویر ۵.۹: (a)) تصویر عکس ۵.۷. d که با فیلتر منظم که NOISEPOWER آن معادل ۴ است به حالت اصلی بازگردانده شده است. (b) همان تصویر با قدرت پارازیت ۰.۴ و حیطه ۷ به حالت اول بازگردانده شده است.

در اینجا PSF ,g از مثال ۵.۷ هستند. تصویر نسبت به حالت اصلی تا حدی بهینه‌سازی شده است. ولی این مقدار خوبی برای NOISEPOWER نیست. پس از آزمایش با این پارامتر و پارامتر RANGE به نتیجه تصویر ۵.۹.(b) می‌رسیم که با استفاده از فرمان زیر حاصل شد:

```
>> fr = deconvreg(g, PSF, 0.4, [1e-7 1e7]);
```

در اینجا می‌بینیم که NOISEPOWER باید یک واحد کم می‌شد و مقدار RANGE نیز باید کمتر از مقدار پیش فرض می‌بود. استفاده از فیلتر Weiner در تصویر ۵.۸. d بهتر است ولی این نتیجه را با داشتن دانش کافی از پارازیت و طیفهای تصویر به دست آوردیم. بدون این اطلاعات نتایج به دست آمده با آزمایش دو فیلتر اغلب قابل قیاس هستند.

اگر تصویر تعمیر شده حلقه‌های تبدیل متمایز فوریر در الگوریتم را نشان دهد، بهتر است قبل از فراخوانی deconvreg از تابع edgetaper (بخش ۵.۷) استفاده کنیم.

۵.۹ مسئله بازگرداندن تصویر به حالت اصلی با روش خطی تکراری و استفاده از الگوریتم لوسی ریچاردسون

Interactive Nonlinear Restoration using the Lucy_Richargson Algorithm

روشهای مسئله بازگرداندن تصویر به حالت اصلی که در ۳ بخش قبل تشریح شدند همگی خطی هستند. از آن جهت مستقیم هستند که وقتی فیلتر تعمیری مشخص می‌شود راه حل با یک دفعه اجرای فیلتر به دست می‌آید. سهولت استفاده و نیاز به محاسبات اندک و داشتن مبانی فرضی محکم باعث شدند که شگردهای خطی ابزار اصلی برای مسئله بازگرداندن تصویر به حالت اصلی در مدت طولانی باشند.

شگردهای تکراری غیرخطی در طی ۲ دهه گشته به صورت ابزار مسئله بازگرداندن تصویر به حالت اصلی پذیرفته شده‌اند و نتایج آنها بهتر از نتایج روشهای خطی است. عیبهای اصلی روشهای غیرخطی آن است که رفتار آنها همیشه قابل پیش بینی نیست و معمولاً به منابع محاسباتی مهم نیاز دارند. اهمیت عیب اول به این دلیل از بین می‌رود که ثابت شده است که روشهای غیرخطی در طیفهای گسترده به شگردهای خطی برتری دارند. افزایش قدرت محاسبات طی دهه اخیر باعث کم رنگ شدن مسئله دوم شده است. روش غیرخطی مورد استفاده در جعبه ابزار شگرد ابداع شده توسط Lucy_Richargson حین کار مستقل بوده است. این روش در جعبه ابزار موسوم به الگوریتم لوسی ریچاردسون است. گاهی وقتها آن را به اسم الگوریتم ریچردسون لوسی نیز می‌بینیم.

ریشه الگوریتم L_R فرمولی با احتمال زیاد است (بخش ۵.۱۰) که تصویر با خصوصیات آماری Poisson الگوبرداری می‌شود. افزایش احتمال وقوع تابع این الگو باعث ایجاد معادله‌ای می‌شود که وقتی تکرار زیر همگرایی می‌یابد حل می‌شود.

$$\hat{f}_{k+1}(x,y) = \hat{f}_k(x,y)[h(-x,-y) * \frac{g(x,y)}{h(x,y) * \hat{f}_k(x,y)}]$$

* نمایانگر تلفیق است. F تخمین تصویر تعمیر نشده است، و g, h در بخش ۵.۱ تعریف شده است. ماهیت تکراری الگوریتم مشخص است. ریشه ماهیت غیرخطی آن از تقسیم f در سمت راست معادله حاصل می‌شود.

مشخص کردن زمان توقف الگوریتم L_R همچون اکثر روشهای غیرخطی مشکل است. در این روش داده‌های خروجی را مشاهده می‌کنیم و وقتی نتیجه قابل قبول در یک الگوریتم به دست می‌آید الگوریتم متوقف می‌شود.

الگوریتم L_R با تابع deconvlucy در IPT اجرا می‌شود که ترکیب زیر را دارد:

```
fr = deconvlucy(g, PSF, NUMIT, DAMPAR, WEIGHT)
```

در اینجا fr تصویر تعمیر شده است.g تصویر مستهلک شده است. PSF تابع گسترش نقاط است. NUMIT تعداد تکرارها است، (مقدار پیش فرض آن ۱۰ است)، و DAMPER, WEIGHT به شکل زیر تعریف می‌شوند.

DAMPER یک کمیت عددی است که انحراف آستانه تصویر حاصل را از تصویر g مشخص می‌کند. جلوی تکرار عناصر تصویری که DAMPER آنها با مقدار اصلی فرق دارد گرفته می‌شود. بنا بر این جلوی تولید پارازیت در این عناصر تصویری گرفته می‌شود، تا مقادیر پیش فرض تصویر حفظ شود. این پیش فرض در حالت بدون استهلاک صفر است.

WEIGHT آرایه‌ای به اندازه g است و برای نشان داده کیفیت هر عنصر تصویری مقداری به آن تخصیص داده می‌شود. مثلاً با تخصیص دادن مقدار صفر به عناصر تصویری نامطلوب می‌توان آنها را از راه حل حذف کرد. روش مفید دیگر برای استفاده از این آرایه تعدیل مقدار عناصر تصویری بر حسب مقدار تصحیح است که ممکن است بر اساس دانش ارایه‌های تصویری ضروری باشد. حین شبیه سازی حالتهای تار با یک PSF خاص (مثال ۵.۷) می‌توان برای حذف عناصر تصویری حاشیه تصویر از محاسبات از WEIGHT استفاده کرد و بنا بر این با PSF به شکلی متفاوت تار می‌شوند. اگر PSF به اندازه n*n* باشد، حاشیه صفرهای مورد استفاده در WEIGHT به عرض ceil n/2 است. مقدار پیش فرض آرایه‌ای واحد به همان اندازه تصویر جی است.

اگر تصویر تعمیر شده حلقه‌های تبدیل متمایز فوریر در الگوریتم را داشته باشد بهتر است قبل از فراخوانی deconvlucy از تابع edgetaper (بخش ۵.۷) استفاده کنیم.

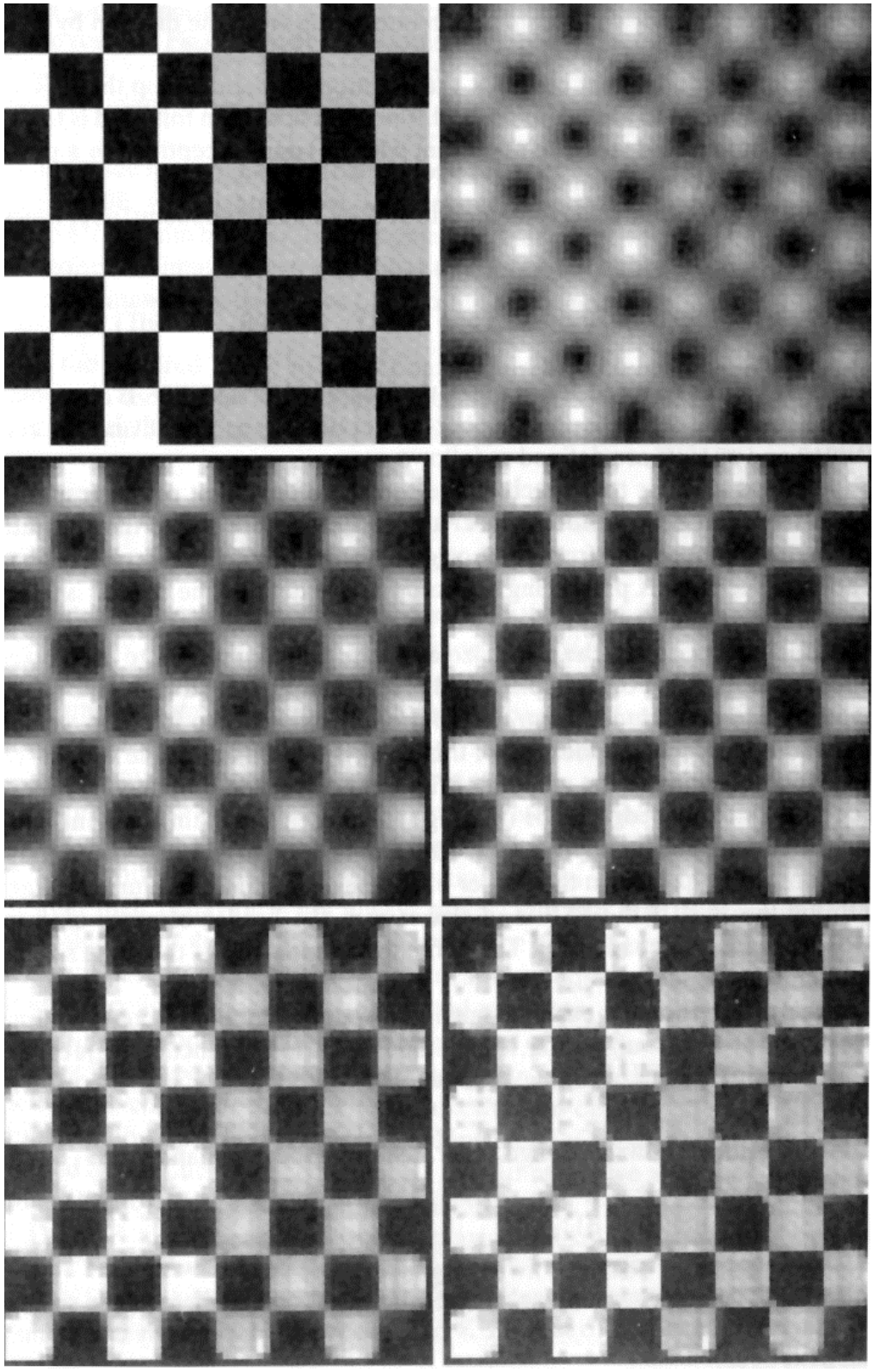
مثال ۵.۱۰: استفاده از تابع deconvlucy برای تعمیر تصویر پارازیت‌دار تار

تصویر ۵.۱۰،(a) تصویر ایجاد شده با استفاده از فرمان زیر را نشان می‌دهد:

```
>> f = checkboard(8);
```

که تصویری مربع شکل به ابعاد ۶۴*۶۴ پیکسل ایجاد کرده است. اندازه تصویر با استفاده از تابع pixeldup به اندازه ۵۱۲*۵۱۲ بزرگ شد تا به خوبی نمایش داده شود.

```
>> imshow(pixeldup(f, 8));
```



تصویر ۵.۱۰: (a)) تصویر اصلی (b) تصویر تار و تحریف شده با پارازیت گوسی (c) تصویر بازگردانده شده به حالت اصلی با استفاده از الگوریتم L_R به ترتیب با تکرارهای ۵.۱۰.۲۰ و ۱۰۰

فرمان زیر یک PSF گوسی به اندازه ۷*۷ با انحراف معیار ۱۰ ایجاد کرد.

```
>> PSF = fspecial('gaussian', 7, 10);
```

سپس تصویر f را با PDF تار کردیم و پارازیت گوسی با میانگین صفر و انحراف معیار ۰.۰۱ را به آن افزودیم.

```
>> SD = 0.01;
>> g = imnoise(imfilter(f,PSF), 'gaussian', 0, SD^2);
```

نتیجه در تصویر ۵.۱۰.(b) نشان داده شده است. مابقی این مثال مربوط به مسئله بازگرداندن تصویر جی به حالت اصلی با استفاده از تابع deconvlucy است. مقدار DAMPER, ۱۰ برابر SD است.

```
>> DAMPAR = 10*SD;
```

آرایه WEIGHT با استفاده از روش تشریح این پارامتر ایجاد شد.

```
>> LIM = ceil(size(PSF, 1)/2);
>> WEIGHT = zero(size(g));
>> WEIGHT(LIM + 1:end – LIM, LIM + 1:end – LIM) = 1;
```

آرایه WEIGHT به اندازه ۶۴*۶۴ است و عرض حاشیه آن 0s چهار پیکسل است. مابقی پیکسلها یک هستند.

تنها متغیر باقیمانده NUMIT یعنی تعداد تکرارها است. نتیجه به دست آمده با استفاده از فرمان زیر در تصویر ۵.۱۰.(c) نشان داده شده است.

```
>> NUMT = 5;
>> fr = deconvlucy(g, PSF, NUMIT, DAMPAR, WEIGHT);
>> imshow(pixeldup(fr, 8))
```

گرچه تصویر بهینه‌سازی شده است ولی هنوز تار است. نتایج به دست آمده با استفاده از numit=10, 20 در تصویرهای ۵.۱۰. d و e نشان داده شده است. این نتیجه مسئله بازگرداندن تصویر پارازیت‌دار تار به حالت اصلی است. در واقع افزایش تعداد تکرارها باعث بهبود زیاد نتیجه نهایی نشد. مثلاً تصویر ۵.۱۰. f با استفاده از ۱۰۰ تکرار به دست آمد. این تصویر اندکی واضح‌تر و روشنتر از تصویر به دست آمده با استفاده از ۲۰ دفعه تکرار است. حاشیه مشکی نازک که در همه نتایج دیده می‌شود توسط صفرهای آرایه WEIGHT ایجاد شد.

۵.۱۰. تلفیق زدایی کورکورانه(Blind Deconvolution)

یکی از مسائل مشکل در مسئله بازگرداندن تصویر به حالت اصلی به دست آوردن مقدار مناسب PSF برای استفاده در الگوریتمهای مسئله بازگرداندن تصویر به حالت اصلی همچون موارد بحث شده در بخش قبل است. همان طور که قبلاً گفته شد آن دسته از روشهای تعمیر تصویر که بر اساس PSF نیستند الگوریتمهای تلفیق‌زدایی کورکورانه نامیده می‌شوند.

یکی از روشهای مهم تلفیق‌زدایی کورکورانه طی ۲ دهه اخیر بر اساس تخمین بزرگترین احتمالات(MLE) است. این راه کار برای برآورد مقادیری است که با پارازیت دلخواه تحریف شده‌اند. برای تفسیر (MLE) می‌گوییم داده‌های تصویری مقادیری دلخواه هستند که از یک سری مقادیر دلخواه دیگر تولید شده‌اند. این احتمالات بر حسب g(x, y), f(x, y), h(x, y) بیان می‌شوند (به بخش ۵.۱ مراجعه کنید). مسئله بعد یافتن حداکثر احتمال وقوع این تابع است. در تلفیق‌زدایی کورکورانه مسئله بهینه‌سازی با تکرار مقادیر محدود مشخص حل می‌شود و همگرایی با توجه به f(x, y), h(x, y) منجر به ایجاد حداکثر مقدار می‌شود که تصویر تعمیر شده و PSF هستند..

```
[fr, PSFe] = deconvblind(g, INITPSF)
```

در اینجا g تصویر مستهلک شده است. Init PSF برآورد مقدماتی تابع گسترش نقطه است. PSFe برآورد نهایی محاسبه شده این تابع است. Fr تصویر تعمیر شده با استفاده از PSF است. الگوریتم مورد استفاده برای به دست آوردن تصویر تعمیر شده الگوریتم تکراری ال – آر است که در بخش ۵.۹ شرح داده شد. برآورد PSF تحت تاثیر اندازه تخمین اولیه قرار می‌گیردو کمتر تحت تاثیر مقادیر است (آرایه یکها نقطه‌ای معقول برای شروع تخمین زدن است).

تعداد دفعات تکرار ترکیب قبل به طور پیش فرض ۱۰ است. برای کنترل تعداد دفعات تکرار و سایر خصوصیات مسئله بازگرداندن تصویر به حالت اصلی همچون ترکیب زیر پارامترهای دیگری نیز دخیل می‌شوند.

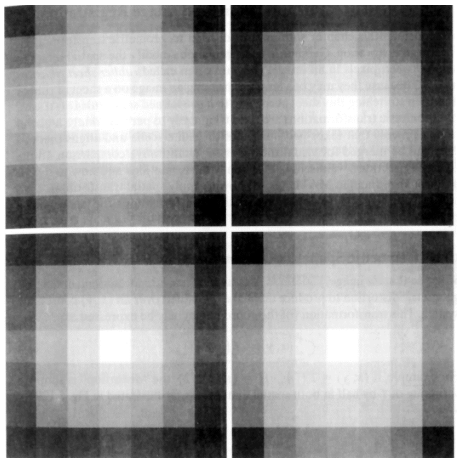
```
[fr, PSFe] = deconvblind(g, INITPSF, NUMIT, DAMPAR, WEIGHT)
```

در اینجا numit, DAMPER, WEIGHT همان مواردی هستند که در الگوریتم L_R در بخش قبل تشریح شده است.

اگر تصویر تعمیر شده حلقه‌های تبدیل متمایز فوریر در الگوریتم را داشته باشد، بهتر است قبل از فراخوانی deconvblind از تابع edgetaper (بخش ۵.۷) استفاده کنیم.

مثال ۵.۱۱. استفاده از تابع deconvblind برای تخمین PSF :تصویر ۵.۱۱.(a) PSF مورد استفاده برای تصویر مستهلک شده در عکس ۵.۱۰.(b) است.

```
>> PSF = fspecial ('gaussian', 7, 10);
>> imshow(pixeldup(PSF, 73), [ ]);
```



تصویر ۵.۱۱. (a)) pse اصلی تخمین PSF با استفاده از ۵، ۱۰ و ۲۰ دفعه تکرار در تابع deconvblind

تصویر مستهلک شده در اینجا همانند مثال ۵.۱۰ با فرمان زیر به دست آورده شد.

```
>> SD = 0.01;
>> g = imnoise(imfilter(f, PSF), 'gaussian', 0, SD^2);
```

در این مثال از تابع deconvblind برای به دست آوردن برآورد PSF با توجه به تصویر مستهلک شده g استفاده می‌کنیم. تصویر PSF(b)۵.۱۱ ناشی از اجرای فرمانهای زیر را نشان می‌دهد:

```
>> INITPSF = ones(size(PSF));
>> NUMIT = 5;
>> [fr, PSFe] = deconvblind(g, INITPSF, NUMIT, DAMPAR, WEIGHT);
>> imshow(pixeldup(PSFe, 73), [ ]);
```

همچون مثال ۵.۱۰ برای DAMPER, WEIGHT از مقادیر یکسان استفاده کردیم.

تصویرهای ۵.۱۱.(c) و d همچون PSFe نشان داده شده است به ترتیب PSFهایی هستند که با ۱۰، و ۲۰ دفعه تکرار به دست آورده شده‌اند. نتیجه دوم نزدیک به PSF واقعی در تصویر ۵.۱۱.(a) است.

۵.۱۱. تبدیلهای هندسی و ثبت تصویر(Geometric Transformation & Image Registration)

این فصل مقدمه‌ای بر تبدیلهای هندسی در مسئله بازگرداندن تصویر به حالت اصلی است. در تبدیلهای هندسی روابط فضایی بین پیکسلهای یک تصویر تغییر می‌کنند. اغلب آنها را تبدیلهای ورقه لاستیکی(rubber_sheet transformation) می‌نامیم چون که ظاهر آنها مانند چاپ یک تصویر روی ورقه لاستیکی است سپس این ورقه بر اساس قوانین از پیش تعیین شده کشیده می‌شود.

برای مسئله بازگرداندن تصویر به حالت اصلی مکرراً از تبدیلهای هندسی استفاده می‌شود. در این فرایند دو تصویر با صحنه‌های یکسان گرفته می‌شود و همراستا می‌شوند تا قابل قیاس و نمایش باشند. مباحث بخش بعد

۱) تبدیلهای فضایی و نحوه تعریف و تجسم آنها در نرم افزار MATLAB

۲) نحوه اجرای تبدیلهای فضایی در تصویرها

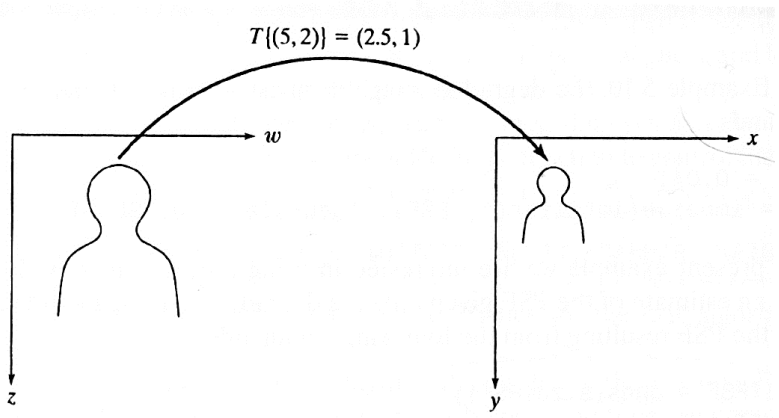
۳) نحوه تعیین تبدیلهای فضایی برای مسئله بازگرداندن تصویر به حالت اصلی است.

۵.۱۱.۱. تبدیلهای فضایی هندسی(Geometric Spatial Transformation)

فرض کنید تصویر f که در حیطه مختصات a(w, z) تعریف شده است دچار تغییر شکل هندسی می‌شود تا تصویر g با مختصات (x, y) تولید شود. این تبدیل مختصات به شکل زیر بیان می‌شود:

```
(x,y) = T{(w,z)}
```

اگر $(x, y) = T\{(w, v)\}=(w/2,z/2)$ باشد این تغییر شکل نتیجه کاهش اندازه f در هر ۲ بعد فضایی است که در تصویر ۵.۱۲ دیده می‌شود.



تصویر ۵.۱۲: یک تبدیل فضایی ساده (توجه داشته باشید که محورهای ایکس-ایگرگ در این تصویر مطابق با مختصات محور تصویر تعریف شده در بخش ۲.۱.۱ نیست). همان طور که در آن بخش گفته شد در این مورد IPT از سیستم مختصات فضایی استفاده می‌کند که ایگرگ نمایانگر ردیفها و ایکس نمایانگر ستونها باشد. این سیستم در این بخش به کار برده شده تا با مطالب IPT در خصوص تبدیلهای هندسی سازگار باشد.

یکی از متداول ترین تبدیلهای فضایی تبدیل affine است. تبدیل affine را می‌توان به شکل ماتریس زیر نوشت:

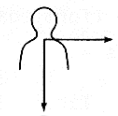
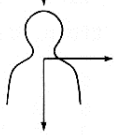
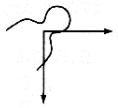
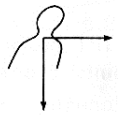
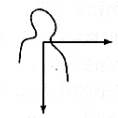
$$[x \quad y \quad 1] = [w \quad z \quad 1]T = [w \quad z \quad 1] = \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

این تبدیل می‌تواند یک مجموعه نقاط را با توجه به مقادیر منتخب عناصر T مقیاس بندی، چرخش، جابجایی و یا برش دهد. در جدول ۵.۳ می‌بینید که برای ایجاد تبدیلهای گوناگون چه مقادیری از عناصر باید انتخاب شود.

IPT نمایانگر تبدیل فضایی با استفاده از ساختارT فرم است.یک راه ایجاد چنین ساختاری استفاده از تابع maketform است که ترکیب آن به شرح زیر است:

```
tform = maketform(transform_type, transform_parameters)
```

جدول ۵.۳: انواع تبدیلهای affine

Type	Affine Matrix, T	Coordinate Equations	Diagram
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = w$ $y = z$	
Scaling	$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = s_x w$ $y = s_y z$	
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = w\cos\theta - z\sin\theta$ $y = w\sin\theta + z\cos\theta$	
Shear (horizontal)	$\begin{bmatrix} 1 & 0 & 0 \\ \alpha & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = w + \alpha z$ $y = z$	
Shear (vertical)	$\begin{bmatrix} 1 & \beta & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = w$ $y = \beta w + z$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \delta_x & \delta_y & 1 \end{bmatrix}$	$x = w + \delta_x$ $y = z + \delta_y$	

اولین شناسه ورودی transform-type یکی از رشته‌های زیر است: " affine "، تصویری" "مکعب"(Bpx) , "ترکیبی" , (Composite) "سفارشی"(Custom) , این نوع تبدیلها در جدول ۵.۴ بخش ۵.۱۱.۳ نشان داده شده است. سایر شناسه‌ها بستگی به نوع تبدیلها دارد و به طور مفصل در راهنمای maketform تشریح شده است.

در این بخش به تبدیلهای affine می‌پردازیم. مثلاً یک راه ایجاد affine tform ایجاد مستقیم ماتریس T از طریق روابط زیر است:

```
>> T = [2  0  0;  0  3  0;  0  0  1];
>> tform = maketform('affine', T)
tform =
      ndims_in:   2
      ndims_out:  2
      forward_fcn:      @fwd_affine
      inverse_fcn:      @inv_affine
 1 struct]×[1             tdata:
```

برای اجرای آن نیازی نیست دامنه‌های ساختار tform مستقیماً اعمال شوند. اطلاعات مربوط به T,T^{-1} در دامنه داده‌های t درج شده است.

```
>> tform.tdata
ans =
      T:    [3 × 3 double]
      Tinv: [3 × 3 double]
>> tform.tdata.Tinv
ans =
      0.5000            0           0
      0           0.3333            0
      0           0           1.0000
```

IPT برای تبدیلهای فضایی ۲ تابع دارد :

۱) tformfwd تبدیل پیشروی $T\{(w, z)\}$ را محاسبه می‌کند .

۲) tforminv تبدیل معکوس $T^{-1}\{(w, z)\}$را محاسبه می‌کند. ترکیب فراخوان xy=tformfwd (wz,tform) است در اینجا wz ماتریسی با p*2 نقطه است. هر ردیف wx حاوی مختصات z, w یک نقطه است. XY ماتریس p*2 است هر ردیف حاوی مختصات ایکس و ایگرگ در یک نقطه تبدیل شده است. مثلاً فرمانهای زیر تبدیل پیشروی یک جفت نقطه و پس از آن تبدیل معکوس را برای تایید اخذ اطلاعات اصلی نشان می‌دهد.

```
>> WZ = [1 1; 3 2];
>> XY = tformfwd(WZ, tform)
XY =
      2     3
      6     6
>> WZ2 = tforminv(XY, tform)
WZ2 =
      1     1
      3     2
```

برای درک آثار یک تبدیل فضایی خاص بهتر است ببینیم که چگونه یک مجموعه نقطه را روی شبکه تبدیل می‌کند. تابع M زیر vistformfwd یک مجموعه شبکه ایجاد می‌کند شبکه را با استفاده از tformfwd تبدیل می‌کند و بعد آن را ترسیم می‌کند سپس شبکه‌های تبدیل شده را در کنار یکدیگر برای مقایسه قرار می‌دهد. به کاربرد توام تابعهای ,meshgrid(بخش ۲.۱۰.۴) و linspace (بخش ۲.۸.۱) برای ایجاد شبکه توجه کنید. کد زیر کاربرد برخی از تابعهای بحث شده تا کنون را نشان می‌دهد.

```
function vistformfwd(tform, wdata, zdata, N)
%VISITFORMFWD Visualize forward geometric transform.
%     VISITFORMFWD(TFORM, WRANGE, ZRANGE, N) shows two plots: an N-by-N
%     grid in the W-Z coordinate system, and the spatially transformed
%     grid in the X-Y coordinate system, WRANGE and ZRANGE are
%     two-element vectors specifying the desired range for the grid. N
%     can be omitted, in which case the default value is 10.

if nargin < 4
      N = 10;
end

% Create the W-Z grid and transform it.
```

```
[W, Z] = meshgrid(linspace(wdata(1), zdata(2), N), …
                                linspace(wdata(1), zdata(2), N);
WZ = [W(:) Z(:) ];
XY = tformfwd([W(:) Z(:) ], tform);

% Calculate the minimum and maximum values of W and x,
% as well as Z and Y. These are used so the two plots can be
% displayed using the same scale.
x = reshape(xy(:, 1), size(W));   % reshape is discussed in Sec. 8.2.2.
y = reshape(xy(:, 2), size(Z));
wx = [w(:); x(:)];
wxlimits = [min(wx) max(wx)];
zy = [z(:); y(:)];
zylimits = [min(zy) max(zy)];

% Create the w-z plot.
Subplot(1, 2, 1) % See Section 7.2.1 for a discussion of this function.
Plot(w, z, 'b'), axis equal, axis ij
Hold on
Plot(w', z', 'b')
Hold off
xlim(wxlimits)
ylim(zylimits)
set(gca, 'XAxisLocation', 'top')
xlabel('w'), ylabel('z')
% Creat the x-y plot.
Subplot(1, 2, 2)
Plot(x, y, 'b'), axis equal, axis ij
Hold on
Plot(x', y', 'b')
Hold off
xlim(wxlimits)
ylim(zylimits)
set(gca, 'XAxisLocation', 'top')
xlabel('x'), ylabel('y')
```

مثال ۵.۱۲. تجسم تبدیلهای affine با استفاده از vistformfwd :

در این مثال از vistformfwd برای تجسم تاثیر چندین نوع تبدیل affine استفاده می‌کنیم. همچنین یک روش دیگر برای ایجاد affine tform با استفاده از maketform. ابتدا کار را با تبدیل affine شروع می‌کنیم که مقیاس افقی آن ۳ و مقیاس عمودی آن ۲ است.

```
>> T1 = [3 0 0; 0 2 0; 0 0 1];
>> tform1 = maketform('affine', T1);
>> vistformfwd(tform1, [0 100], [0 100]);
```

نتیجه در تصویرهای ۵.۱۳.(a) و (b) نشان داده شده است.

وقتی t_{12}, t_{21} در ماتریس تی affine زیر صفر نباشند تاثیر برشی روی می‌دهد:

```
>> T2 = [1 0 0; .2 1 0; 0 0 1];
>> tform2 = maketform('affine', T2);
>> vistformfwd(tform2, [0 100], [0 100]);
```

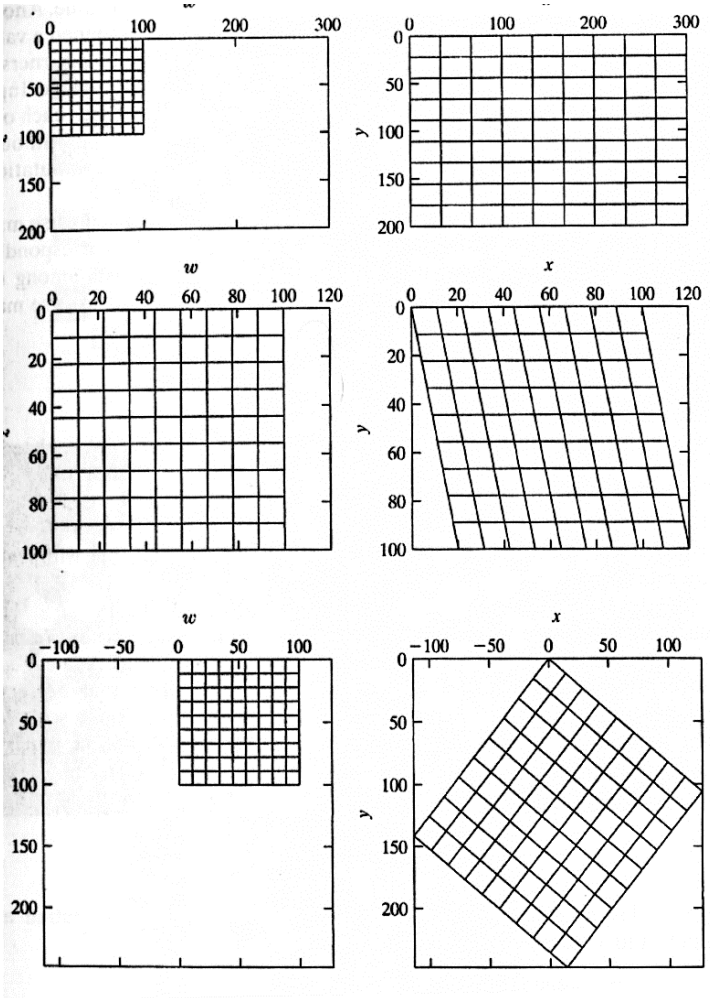
تاثیر تبدیل برشی روی شبکه در تصویرهای ۵.۱۳(c) و d نشان داده شده است.

یک خصوصیت جالب تبدیل affine آن است که ترکیب چند تبدیل affine می‌تواند یک تبدیل affine شود. یعنی تبدیلهای affine را می‌توان با استفاده از ضرب ماتریسهای T ایجاد کرد. بلوک رمزهای بعدی نحوه ایجاد و تجسم تبدیل affine است که ترکیبی از مقیاس بندی، چرخش و برش است.

```
>> Tscale = [1.5 0 0; 0 2 0; 0 0 1];
>> Trotation = [cos(pi/4)  sin(pi/4)  0
              -sin(pi/4)  cos(pi/4)  0
              0 0 1];

>> Tshear = [1 0 0; .2 1 0; 0 0 1];
>> T3= Tscale * Trotation * Tshear;
>> tform3 = maketform('affine', T3);
>> vistformfwd(tform3, [0 100], [0 100])
```

نتایج در تصویرهای ۵.۱۳e و f نشان داده شده است.



۵.۱۱.۲. اجرای تبدیلهای فضایی روی تصویرها(Applying Spatial Transformation to Image)

اکثر روشهای محاسباتی تبدیل فضایی تصویرها به ۲ مقوله تقسیم می‌شوند: روشهای استفاده از نمایش پیشرو(forward mapping) و روشهای ترسیم معکوس(inverse mapping). در روش ترسیم پیشرو هر یک از نقاط ورودی اسکن می‌شود و مقدار آن به داده‌های خروجی تصویر در موضعی که با $T\{(w, z)\}$ مشخص می‌شود کپی می‌شود. یکی از مسائل ترسیم پیشرو آن است که ۲ یا چند پیکسل در تصویر ورودی را می‌توان به همان تعداد پیکسل در تصویر خروجی تبدیل کرد. به این ترتیب، مشخص می‌شود که چگونه مقادیر چندین تصویر ورودی به یک تصویر خروجی تبدیل می‌شود. مسئله دیگر آن است که ممکن است به برخی پیکسلهای خروجی هیچ مقداری تخصیص داده نشود. در حالت ترسیم پیشروی پیشرفته ۴ گوشه هر پیکسل ورودی روی چهارگوشهای تصویر خروجی ترسیم می‌شوند. توزیع پیکسلهای ورودی در بین پیکسلهای خروجی بر اساس گستره پیکسلهای خروجی است که متناسب با ناحیه هر پیکسل خروجی است. گرچه این نوع ترسیم پیشرو دقیقتر است ولی پیچیده است و اجرای محاسبات آن گران است.

تابع IPT imtransform در عوض از ترسیم معکوس استفاده می‌کند. در روش ترسیم معکوس هر عنصر تصویری خروجی پیمایش می‌شود و محل مطابق با آن در تصویر ورودی با استفاده از $T^{-1}\{(w, z)\}$ محاسبه می‌شود، و برای تعیین مقدار پیکسل خروجی در بین نزدیکترین پیکسلهای ورودی درون یابی می‌شود. اجرای ترسیم معکوس راحتتر از ترسیم پیشرو است.

ترکیب فراخوانی imtransform به شرح زیر است:

```
g = imtransform(f, tform, interp)
```

در اینجا interp رشته‌ای است که نشان می‌دهد چگونه پیکسلهای تصویر ورودی برای به دست آوردن پیکسلهای خروجی درون یابی می‌شوند: این درون یابی ممکن است به صورت نزدیکnearest ، دو خطیbilinear و یا دو مکعبیbicubic باشد. می‌توان شناسه درون یابی را حذف کرد که در این صورت مقدار پیش فرض آن همانند حالت دو خطی می‌شود. همان طور که در مثالهای تعمیر در قبل گفتیم تابع تخته شطرنجی برای ایجاد تصویرهای آزمایشی برای آزمایش روی تبدیلهای فضایی انجام می‌شود.

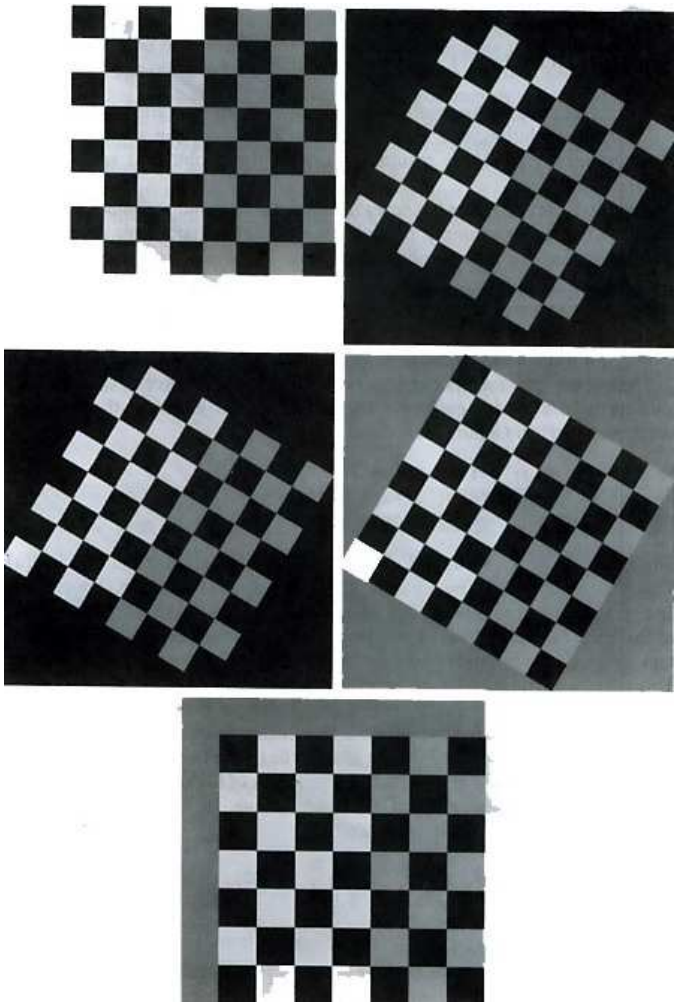
در این مثال از تابعهای checkerboard, imtransform برای کشف جنبه‌های مختلف تصویر تبدیل شده استفاده می‌کنیم. تبدیل همشکل خطی(linear conformal Transformation) نوعی تبدیل affine است که شکلها و زاویه‌ها را حفظ می‌کند. تبدیلهای همشکل خطی از یک عامل مقیاس بندی، زاویه چرخش، و عامل جابجایی تشکیل می‌شوند. ماتریس تبدیل affine در این حالت به شکل زیر است:

$$T = \begin{bmatrix} s\cos\theta & s\sin\theta & 0 \\ -s\sin\theta & s\cos\theta & 0 \\ \delta_x & \delta_y & 1 \end{bmatrix}$$

فرمانهای زیر یک تبدیل همشکل خطی ایجاد می‌کنند و آن را روی تصویر آزمایشی امتحان می‌کنند.

```
>> f = checkerboard(50);
>> s = 0.8;
>> theta = pi/6;
>> T = [s*cos(theta) s*sin(theta)  0
           - s*sin(theta) s*cos(theta)  0
           0  0  1];
>> tform = maketform('affine', T);
>> g = imtransform(f, tform);
```

تصویرهای صفحه شطرنجی تبدیل شده و اصلی در عکسهای ۵.۱۴.(a) و (b) نشان داده شده است. در فراخوانی imtransform از روش درون یابی پیش فرض استفاده شده است.



تصویر ۵.۱۴. تبدیلهای affine تصویر شطرنجی (a)) تصویر اصلی تبدیل همشکل خطی با استفاده از درون یابی پیش فرض (دو خطی) (c) استفاده از نزدیکترین درون یابی همجوار ت) مشخص کردن مقدار لازم برای پر کردن (e) کنترل فضای خروجی طوری که جابجایی قابل رویت باشد.

همان طور که قبلاً گفتیم می‌توان یک روش درون یابی متفاوت انتخاب کرد. مثلاً نزدیکترین نقطه همجوار با فراخوانی تابع imtransform

```
>> g2 = imtransform(f, tform, 'nearest');
```

نتیجه در تصویر ۵.۱۴.cنشان داده شده است. درون یابی نزدیکترین نقطه همجوار سریع‌تر از درون یابی دو خطی است و در برخی مواقع می‌تواند مناسبتر باشد. ولی معمولاً نتایجی ایجاد می‌کند که پستتر از نتایج به دست آمده از درون یابی دو خطی است.

تابع imtransform چندین پارامتر اضافی دیگر دارد که در برخی مواقع مفید هستند. مثلاً اگر پارامتر fillvalue در آن قرار داده شود رنگی را که imtransform برای پیکسلهای خارج از دامنه تصویر ورودی استفاده می‌کند کنترل می‌کند.

```
>> g3 = imtransform(f, tform, 'FillValue', 0.5);
```

در تصویر ۵.۱۴. d پیکسلهای خارج از تصویر اصلی به جای مشکی نیمه خاکستری هستند.

برای حل مسائل مربوط به جابجایی تصاویر با استفاده از imtransform می‌توان از پارامترهای دیگر نیز استفاده کرد. مثلاً نتایج زیر باعث ارائه جابجایی‌های خالص می‌شود:

```
>> T2= [1 0 0; 0 1 0; 50 50 1];
>> tform2 = maketform('affine', T2);
>> g = imtransform(f, tform2);
```

نتایج شبیه به تصویر اصلی عکس ۵.۱۴.(a) اس. این تاثیر در رفتار پیش فرض imtransform دیده می‌شود. Imtransform کادر تعیین شده را تعریف می‌کند. (برای تعریف عبارت مکعب کراندار به بخش ۱۱.۴.۱ مراجعه کنید) و تصویر خروجی در سیستم مختصات بیرونی قرار داده می‌شود. فقط ترسیم معکوس در مکعب کراندار انجام می‌شود. این عامل باعث لغو جابجایی می‌شود. با مشخص کردن پارامترهای xdata, ydata می‌توان دقیقاً برای imtransform مشخص کرد که در کام قسمت از فضای خروجی نتیجه را محاسبه کند. Xdata یک بردار ۲ عنصری است که موقعیت ستونهای چپ و راست تصویر خروجی را مشخص می‌کند. Ydata یک بردار ۲ عنصری است که موقعیت ردیفهای بالا و پایین تصویر خروجی را مشخص می‌کند. فرمان زیر تصویر خروجی را در حیطه بین $(x, y) = (1, 1)$ و $(x, y) = (400, 400)$ محاسبه می‌کند.

```
>> g5 = imtransform(f, tform2, 'XData', [1 400], 'YData', [1 400], …
                                'FillValue', 0.5);
```

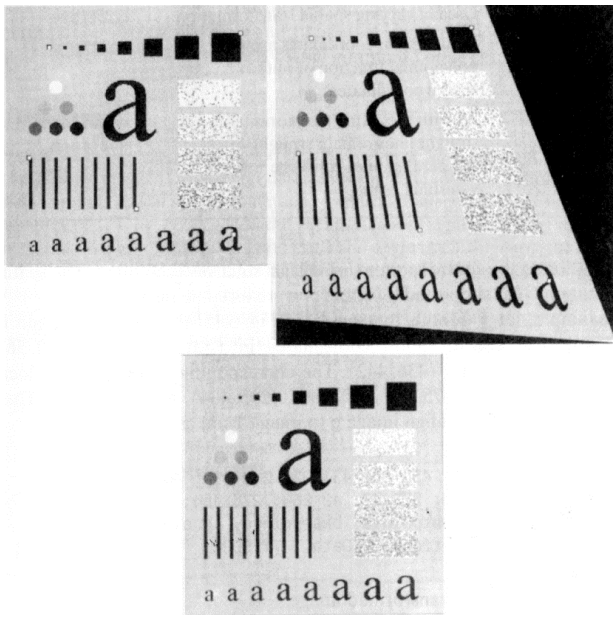
نتیجه در تصویر ۵.۱۴ e نشان داده شده است.

سایر تنظیمات imtransform و تابعهای مرتبط IPT باعث کنترل بیشتر نتیجه می‌شود مخصوصاً نحوه اجرای درون یابی. اکثر اطلاعات مربوط به جعبه ابزار در صفحات راهنمای تابعهای imtransform, makeresampler است.

۵.۱۱.۳. ثبت تصویر (Image Registration)

در روشهایی ثبت تصویر دو تصویر هم صحنه همراستا می‌شوند. مثلاً می‌توان دو یا چند تصویر را که همزمان گرفته شده با ابزار گوناگون همچون تصویربرداری تشدید مغناطیسی و اسکن موضع نگاری با ساطع کردن الکترونهای مثبت همراستا کرد. ممکن است تصاویر در زمانهای مختلف با ابزار یکسان همچون تصویربرداری ماهواره‌ای از یک منطقه در روزها، ماهها یا حتی سالهای گوناگون گرفته شده باشند. در هر دو مورد ادغام تصویرها و اجرای تحلیل کمی و مقایسه مستلزم جبران میزان انحراف هندسی ناشی از اختلاف زاویه دوربین، مسافت، جهتگیری، تفکیک پذیری رنگ حسگرها، تغییر موقعیت‌های موضوع، و سایر عوامل است.

این جعبه ابزار برای پشتیبانی از ثبت تصویر از نقاط کنترل بهره می‌برد که آنها را نقاط اتصال نیز می‌نامیم که زیر مجموعه‌ای از پیکسلهایی هستند که موقعیت‌های آنها در دو تصویر شناخته شده است و می‌توان آنها را به صورت محاوره‌ای انتخاب کرد. فکر استفاده از نقاط کنترلی با بکارگیری الگوی آزمایشی و الگویی که تصویر آن تغییر شکل داده است در عکس ۵.۱۵ به تصویر کشیده شده است. پس از انتخاب تعداد کافی نقاط کنترل، از تابع cp2tform IPT برای تطابق یک نوع تبدیل فضایی خاص در نقاط کنترل (با استفاده از شگردهای کوچکترین مربعات) بهره‌برداری می‌شود.



تصویر ۵.۱۵. ثبت تصویر بر اساس نقاط کنترل (a)) تصویر اصلی با نقاط کنترل (دایره‌های کوچک روی تصویر قرار دارند) (b) تصویر تحریف شده هندسی با نقاط کنترل (c) تصویر تصحیح شده با استفاده از تبدیل ناشی از نقاط کنترل

جدول ۵.۴. تبدیلهای مورد پشتیبانی cp2tform, maketform

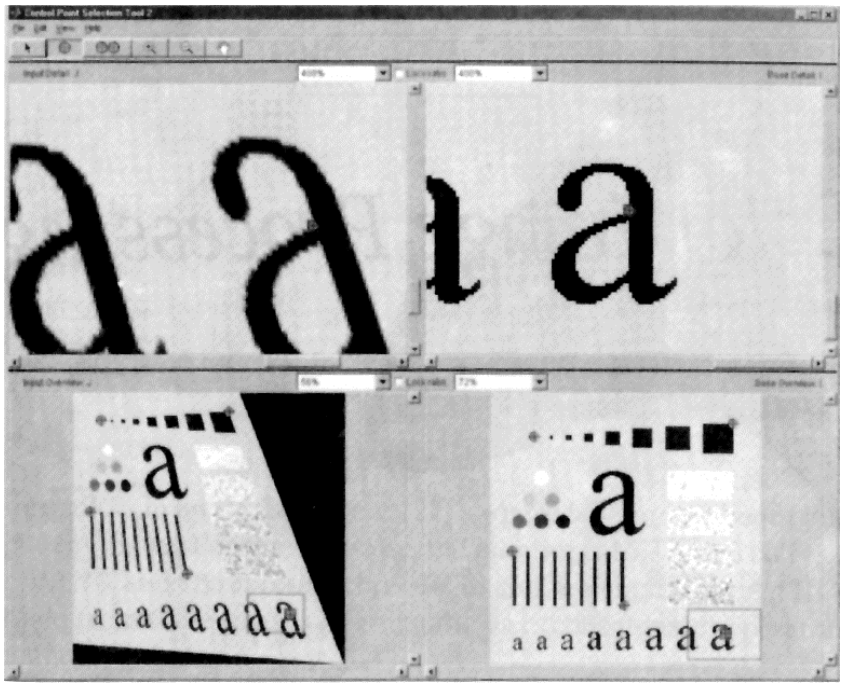
نوع تبدیل	تشریح تبدیل	تابع‌ها
affine	ادغام مقیاس بندی، چرخش، برش و جابجایی. خطوط مستقیم و موازی بدون تغییر باقی می‌مانند.	maketform cp2tform
مکعب Box	مقیاس بندی و جابجایی مستقل در امتداد هر بعد و زیرمجموعه affine	maketform
مرکب Composite	یک مجموعه از تبدیلهای فضایی که پی در پی اعمال می‌شوند.	maketform
سفارشی Custom	تبدیل فضایی سفارشی. کاربر تابعهای تعریف کننده t, t’ را ایجاد می‌کند.	maketform
همشکل و خطی Linear conform	مقیاس بندی (یکسان در همه ابعاد) چرخش و جابجایی در زیرمجموعه‌ای از مواضع affine	cp2tform
میانگین ارزش موضعی LWM	میانگین مقدار موضعی، یک تبدیل فضایی که در هر موضع فرق می‌کند	cp2tform
خطی تکه به تکه Piecewise linear	تبدیلهای فضایی با مواضع گوناگون	cp2tform
چند جمله‌ای Polynomial	مختصات فضایی داده‌های ورودی تابع چند جمله‌ای مختصات فضایی داده‌های خروجی است	cp2tform
تصویری Projective	همان طور که در تبدیل affine دیده می‌شود خطوط مستقیم تغییر نمی‌کنند ولی خطوط موازی تا بینهایت همگرایی دارند.	maketform cp2tform

تبدیلهای فضایی مورد پشتیبانی Cp2tform در جدول ۵.۴ فهرست بندی شده است.

مثلاً اگر f نمایانگر تصویر ۵.۱۵.(a) باشد و جی تصویر عکس ۵.۱۵.(b) باشد، مختصات نقطه کنترلی f (۸۳و ۸۱) (۴۵۰و ۵۶) و (۴۳۶و ۴۴۲) هستند. نقاط کنترل کننده مطابق با آنها در جی (۶۸و ۶۶)، (۲۷۵و ۴۳۴) و (۵۲۳و ۵۳۲) هستند. فرمانهای مورد نیاز برای همراستا کردن تصویر جی در تصویر f به شرح زیر است:

```
>> basepoints = [83 81; 450 56; 43 293; 249 392; 436 442];
>> inputpoints = [68 66; 375 47; 42 286; 275 434; 523 532];
>> tform = cp2tform(inputpoints, basepoints, 'projective');
>> gp = imtransform(g, tform, 'XData', [1 502], 'YData', [1 502]);
```

تصویر تبدیل شده در عکس ۵.۱۵.cنشان داده شده است.



تصویر ۵.۱۶. ابزار محاوره‌ای برای انتخاب نقاط کنترل کننده

این جعبه ابزار یک رابط گرافیکی برای کاربر دارد که برای انتخاب نقاط کنترل کننده در یک جفت عکس است. تصویری از این ابزار در عکس ۵.۱۶ دیده می‌شود که با فرمان cpselect فراخوانی شده است.

خلاصه مطالب(Summary)

در مطالب این فصل نشان دادیم که چگونه تابعهای IPT و نرم افزار MATLAB برای بازگرداندن تصویر به حالت اصلی به کار برده می‌شوند و چگونه مبنای ایجاد الگوهایی هستند که از طریق آنها می‌توان میزان کاهش رنگ تصویر را تشریح کرد. در این فصل با ابداع تابعهای imnoise2, imnoise3 قابلیت‌های IPT برای ایجاد پارازیت تقویت شد. فیلترهای فضایی موجود در تابع spfilt مخصوصاً فیلترهای غیرخطی برای تعمیم قابلیت‌های IPT در این زمینه است. در مثالهای این تابعها نشان دادیم که ادغام تابعهای IPT و نرم افزار MATLAB به کدهای جدید برای ایجاد برنامه‌هایی برای تقویت قابلیت‌های مجموعه گسترده‌ای از ابزار موجود چقدر آسان است.