Доля П.Г.
Харьковский Национальный Университет
механико – математический факультет
2014 г.

Математические методы обработки изображений.

В настоящем пособии излагаются методы, которые используются при обработке изображений. Перед вами часть, которая знакомит с математическими методами восстановления изображений искаженных аддитивным шумом.

Восстановление изображений с шумом.

Оглавление

6. Восстановление изображений с шумом

Задача улучшения изображения заключается в изменении его качества, которое зависит от особенностей человеческого зрения. Целью же восстановления является реконструкция изображения, которое ранее было искажено в результате внешнего воздействия, про которое имеется некоторая информация. Поэтому методы восстановления изображения основаны на моделировании процессов искажения и применении обратных процессов для воссоздания исходного изображения.

Процесс ухудшения изображения моделируется в виде функции искажения, которая вместе с аддитивным шумом действует на исходное изображение $f(x,y)$ и порождает искаженное изображение $g(x,y)$:

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

Имея функцию $g(x,y)$, обладая некоторой информацией об искажающем операторе H и зная основные характеристики аддитивного шума $\eta(x,y)$, можно построить некоторое приближение $\tilde{f}(x,y)$ исходного изображения.

$$\tilde{f}(x,y) = H^{-1}(g(x,y) - \eta(x,y))$$

Поскольку функция шума η(x,y) – случайная и ее нельзя смоделировать абсолютно точно, то возможно только приблизительное восстановление изображения. Поэтому целью восстановления изображения является построение приближения $\tilde{f}(x,y)$, которое было бы максимально близко к исходному изображению. При этом чем больше информации мы имеем об операторе H и шуме $\eta(x,y)$, тем точнее мы сможем приблизить изображение $f(x,y)$ функцией $\tilde{f}(x,y)$. В этой части мы будем предполагать, что H — тождественный оператор, т. е. искажения вносятся исключительно шумом. Позже мы откажемся от этого предположения.

Прежде, чем моделировать случайную функцию шума $\eta(x,y)$, вспомним некоторые основные понятия теории вероятностей.

6.1 Элементы теории вероятностей

Напомним некоторые определения. Если случайному событию поставлено в соответствие некоторое число, то говорят, что задана случайная величина. Когда значения, которые может принимать случайная величина X, является дискретным (конечным или бесконечным) набором чисел $x_1, x_2, \ldots, x_n, \ldots$, то и сама случайная величина называется дискретной. Если же значения, которые может принимать случайная величина X заполняют конечный или бесконечный промежуток (a,b) числовой оси, то случайная величина называется непрерывной.

Каждому значению случайной величины дискретного типа x_n отвечает определенная вероятность p_n; каждому промежутку (a,b) из области значений непрерывной случайной величины также отвечает определенная вероятность $P(a < X < b)$ того, что значение, принятое случайной величиной, попадает в этот промежуток.

Соотношение, устанавливающее тем или иным способом связь между возможными значениями случайной величины и их вероятностями, называется законом распределения случайной величины.

Закон распределения дискретной случайной величины обычно задается рядом распределения

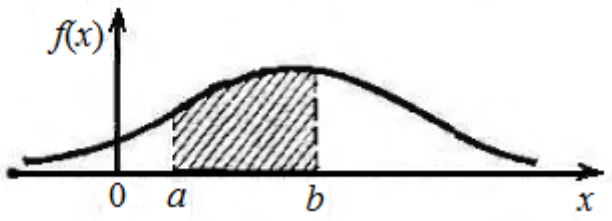
$$\begin{bmatrix} x_i & x_1 & x_2 & x_3 & \ldots & x_n \\ p_i & p_1 & p_2 & p_3 & \ldots & p_n \end{bmatrix}$$

При этом $\sum_i p_i = 1$, и суммирование распространяется на все (конечное или бесконечное) множество значений данной случайной величины X.

Закон распределения непрерывной случайной величины задается с помощью функции плотности вероятности $f(x)$. Вероятность $P(a < X < b)$ того, что значение случайной величины X попадает в промежуток (a, b) определяется равенством

$$P(a < X < b) = \int_a^b f(x)\, d\, x$$

График функции f(x) называется кривой распределения. Геометрически вероятность попадания случайной величины в промежуток (a, b) равна площади криволинейной трапеции, ограниченной сверху кривой распределения, снизу – осью Ox, слева и справа – прямыми $x = a, x = b$.



Функция плотности вероятности $f(x)$ обладает следующими свойствами:

1^0. $f(x) \geq 0$

2^0. $\int_{-\infty}^{\infty} f(x) dx = 1$

Функция $F(x) = P(X < x)$ называется функцией распределения вероятности случайной величины X. Функция $F(x)$ существует как для дискретных, так и для непрерывных случайных величин. Если $f(x)$ – функция плотности распределения вероятности непрерывной случайной величины X, то $F(x) = \int_{-\infty}^{x} f(\xi) d\xi$. Из последнего равенства следует, что $f(x) = \frac{d\, F(x)}{d\, x}$.

Отметим основные свойства функции распределения вероятности:

1^0. $F(x)$ – неубывающая функция.

2^0. $F(-\infty) = 0$,

3^0. $F(+\infty) = 1$.

Математическим ожиданием дискретной случайной величины называется сумма произведений возможных значений случайной величины на вероятности этих значений. Если случайная величина X характеризуется конечным набором значений, то математическое ожидание M(X) определяется по формуле

$$M(X) = \sum_{i=1}^{n} x_i\, p_i \qquad (1)$$

Математическое ожидание для непрерывной случайной величины определяется равенством

$$M(X) = \int_{-\infty}^{+\infty} x\, f(x) d\, x \qquad (2)$$

где $f(x)$ – плотность вероятности случайной величины.

Дисперсией случайной величины называют математическое ожидание квадрата отклонения случайной величины от ее математического ожидания:

$$D(X)=M\left([X-M(X)]^2\right) \tag{3}$$

Если ввести обозначение $M(X)=m$, то формула для вычисления дисперсии дискретной случайной величины X запишется в виде

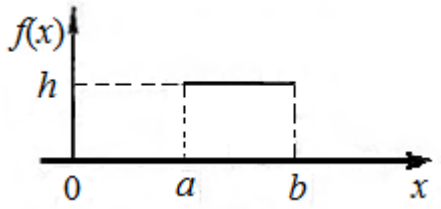
$$D(X)=\sum_i p_i(x_i-m)^2, \tag{4}$$

а для непрерывной случайной величины X – в виде

$$D(X)=\int_{-\infty}^{+\infty}(x-m)^2 f(x)dx \tag{5}$$

Средним квадратичным отклонением случайной величины X называется величина $\sigma_x=\sqrt{D(X)}$. Среднее квадратичное отклонение есть мера рассеяния значений случайной величины около ее математического ожидания.

Равномерным называется распределение таких случайных величин, все значения которых лежат на некотором отрезке $[a,b]$ и имеют постоянную плотность вероятности на этом отрезке, т.е.



$$f(x)=\begin{cases}0, & x<a\\ \dfrac{1}{b-a}, & a\le x\le b\\ 0, & x>b.\end{cases}$$

Закон распределения случайной величины X, которая может принимать любые целые неотрицательные значения (0, 1, 2,...,n,...), описываемый формулой

$$P(X=m)=\frac{a^m}{m!}e^{-a}$$

носит название **закона Пуассона**.

Математическое ожидание и дисперсия случайной величины, распределенной по закону Пуассона, определяются по формулам

$$M(X)=a;\ D(X)=a.$$

Аналогом закона Пуассона для непрерывных случайных величин служит **показательный (экспоненциальный) закон**, функция плотности распределения которого имеет вид $(\lambda>0)$

$$f(x)=\begin{cases}0, & x<0\\ \lambda e^{-\lambda x}, & x\ge 0\end{cases} \tag{6}$$

Функция распределения экспоненциального закона равна (при $x>0$)

$$F(x)=\int_{-\infty}^{x} f(\xi)d\xi=\int_0^x \lambda e^{-\lambda\xi}d\xi=1-e^{-\lambda x} \tag{7}$$

Тогда

$$F(x)=\begin{cases} 0 & , x<0 \\ 1-e^{-\lambda x}, & x\geq 0 \end{cases} \tag{8}$$

Числовые характеристики показательного закона распределения равны следующим значениям. Математическое ожидание

$$M(X)=\int_0^{\infty} x\lambda e^{-\lambda x}\,dx=\left(-xe^{-\lambda x}-\frac{1}{\lambda}e^{-\lambda x}\right)\Big|_0^{\infty}=\frac{1}{\lambda} \tag{9}$$

Дисперсия

$$D(X)=\int_0^{\infty}\left(x-\frac{1}{\lambda}\right)^2\lambda e^{-\lambda x}\,dx=\lambda\int_0^{\infty}\left(x^2-\frac{2}{\lambda}x+\frac{1}{\lambda^2}\right)e^{-\lambda x}\,dx=$$

$$=-\int_0^{\infty}x^2 de^{-\lambda x}-2\int_0^{\infty}x\,e^{-\lambda x}\,dx+\frac{1}{\lambda}\int_0^{\infty}e^{-\lambda x}\,dx=$$

$$=-\left(x^2e^{-\lambda x}\Big|_0^{\infty}-2\int_0^{\infty}x\,e^{-\lambda x}\,dx\right)-2\int_0^{\infty}x\,e^{-\lambda x}\,dx+\frac{1}{\lambda}\int_0^{\infty}e^{-\lambda x}\,dx=\frac{1}{\lambda^2} \tag{10}$$

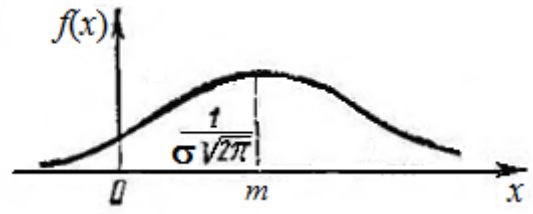
и среднеквадратическое отклонение

$$\sigma(X)=\sqrt{D(X)}=\frac{1}{\lambda} \tag{11}$$

Нормальный закон распределения характеризуется плотностью

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-(x-m)^2/(2\sigma^2)} \tag{13}$$

График этой функции показан на следующем рисунке



Он симметричен относительно прямой $x=m$, максимум достигается при $x=m$ и он равен $1/(\sigma\sqrt{2\pi})$. Поскольку $\int_{-\infty}^{\infty}x\,f(x)dx=m$, то параметр m является математическим ожиданием случайной величины X. Кроме того, поскольку $\int_{-\infty}^{\infty}(x-m)^2\,f(x)dx=\sigma^2$, то $D(x)=\sigma^2$, т.е. σ является средним квадратичным отклонением величины X.

6.2 Модели шума

Возможность смоделировать шум является ключевым моментом восстановления изображений. В этом параграфе рассматриваются два типа моделей шумов: шум в пространственной области (характеризующийся функцией плотности вероятности) и шум в частотной области, который

описывается характеристиками Фурье преобразования. Везде мы полагаем, что шум не зависит от координат пикселя изображения.

Шум в пространственной области мы представляем как некоторые случайные колебания в яркости изображения, т.е. некоторые случайные добавки к яркости каждого пикселя изображения, которые можно трактовать как добавочное изображение. Значения яркостей пикселей этого «случайного» изображения можно рассматривать как случайные величины, характеризующиеся плотностью распределения вероятностей (PDF, Probability Density Function) и функцией распределения (функцией кумулятивного распределения; CDF - Cumulative Distribution Function) .

В Matlab имеется функция `imnoise`, которая моделирует искажение изображения некоторым шумом. Синтаксис этой функции имеет вид

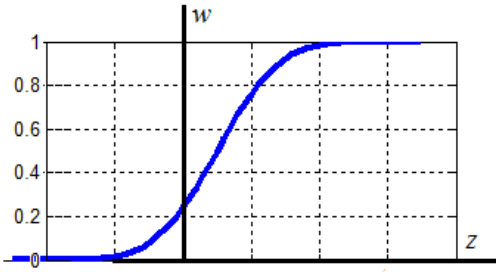
```
g=imnoise(f, type, parameters),
```

где `f` — это исходное изображение, аргумент `type` является строкой, определяющей тип функции распределения вероятности, а аргументы `parameters` определяют параметры этой функции (математическое ожидание и дисперсию). Функция `imnoise` сначала преобразует изображение в класс `double` в диапазоне [0,1] накладывает шум, а затем преобразует изображение обратно в исходный тип. Это следует иметь в виду перед заданием параметров шума. Например, чтобы прибавить шум со средним 32 и дисперсией 200 к изображению класса `uint8`, следует сжать среднее до величины 32/256, а дисперсию приравнять $200/256^2$, после чего эти параметры можно подставлять в функцию `imnoise`. Параметр `type` может принимать следующие значения

`'gaussian'`	Гауссов шум с постоянным средним и дисперсией
`'localvar'`	Гауссов шум с нулевым средним и переменной дисперсией
`'poisson'`	Шум Пуассона
`'salt & pepper'`	Шум «соль и перец»
`'speckle'`	Мультипликативный шум ($J = I + n \cdot I$, где n равномерно распределенный шум)

Иногда может понадобиться генерирование шумов других типов. Тогда можно воспользоваться следующими соображениями.

Пусть имеется случайная величина w с равномерным распределением на отрезке [0, 1]. Для построения случайной величины с заданной функцией распределения $F(z)$ надо решить уравнение $F(z) = w$ относительно z. Это поясняет следующий рисунок



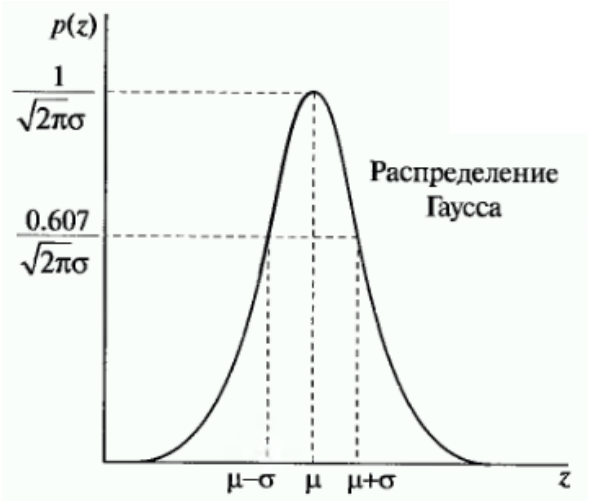
Т.е. случайную величину z надо построить по формуле $z = F^{-1}(w)$. В Matlab матрица случайных величин с равномерным распределением значений на отрезке $[0,1]$ генерируется с помощью функции `rand(M,N)`. Тогда случайный шум размером M x N с функцией распределения вероятности $F(z)$ в MATLAB можно сгенерировать командой

$$z = F^{-1}(\text{rand}(M, N)) \tag{1}$$

Гауссов шум. Яркость пикселей изображения гауссова шума, который накладывается на другое изображение, характеризуется случайной величиной с гауссовой функцией плотности распределения вероятности (по-другому – нормальный закон распределения вероятности) вида (1.13)

$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(z-\mu)^2/2\sigma^2}, \tag{2}$$

где z представляет собой значение яркости, μ - среднее значение случайной величины z, σ – ее среднеквадратическое отклонение. Квадрат среднеквадратического отклонения σ^2 является дисперсией величины z. График этой функции представлен на следующем рисунке. Когда плотность распределения случайной величины z описывается этой функцией, то приблизительно 70% ее значений попадают в диапазон $[\mu-\sigma, \mu+\sigma]$, и примерно 95% - в диапазон $[\mu-2\sigma, \mu+2\sigma]$.



Пример. Построение кривой распределения Гаусса (следующий рисунок слева)

```
m=0.5;          % матожидание
s=0.1;          % среднеквадратичное отклонение
z=0:0.01:2*m;
```

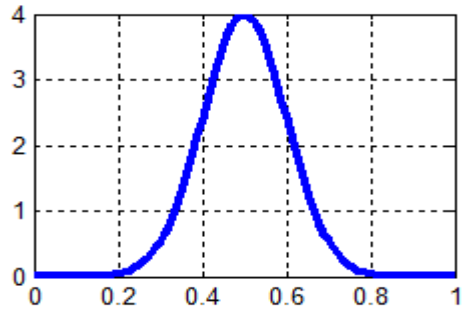
```
p=@(z) 1/(sqrt(2*pi)*s)*exp(-(z-m).^2/(2*s^2));
plot(z,p(z),'LineWidth',3); grid on;  % след. рисунок слева
```

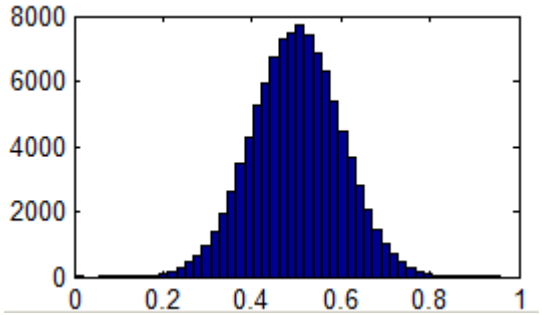
Гауссов шум используется в качестве приближения в тех случаях, когда детекторы изображения работают на пороге чувствительности. Функция Matlab из IPT пакета

```
g=imnoise(I,'gaussian',m,s)
```

добавляет к изображению I гауссов шум со средним m и дисперсией s. По умолчанию m=0 и s=0.01. Вот пример кода построения гистограммы гауссова шума с помощью этой функции

```
I=zeros(1,100000);  % одномерный вектор из нулей
J = imnoise(I,'gaussian',0.5,0.01);  % 0.01 - дисперсия
hist(J,50);  % след. рисунок справа
```

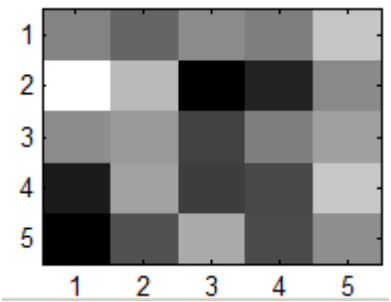




Для пояснения работы функции imnoise построим изображение шума 5 x 5 пикселей.

```
I=zeros(5,5)  % пустое изображение (черные пиксели)
J = imnoise(I,'gaussian', 0.5,0.01)  % 0.01 - дисперсия
J3=cat(3,J,J,J) % создаем 3-х мерный массив серого изображения
imagesc(J3);  % imagesc рассматривает 3-х мерный массив как RGB
```

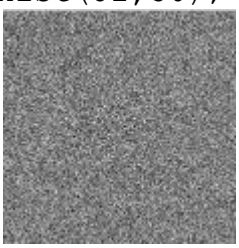
Для того, чтобы функция imagesc(...) воспринимала числовые значения в изображении как яркость пикселей (а не как номера в палитре цветов), изображение должно быть представлено 3-х мерным массивом. Для этого используется функция cat(3,A,B,C,...), которая создает 3-х мерный массив из матриц A,B,C,... одинакового размера.

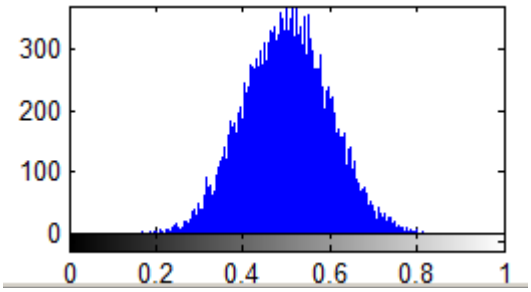


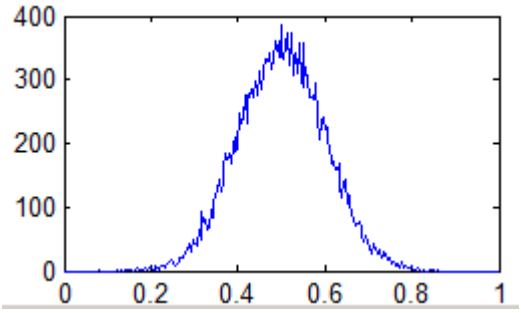
Пример сценария, который строит изображение гауссова шума и его плотность распределения вероятности

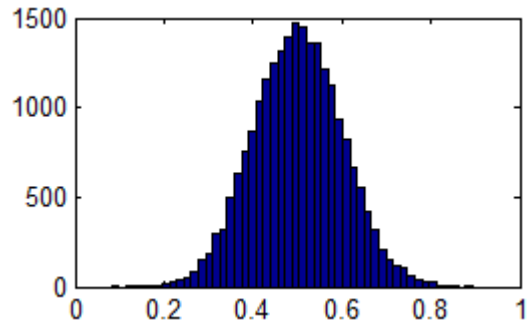
```
close all
m=0.5;   % определяет среднюю яркость изображения шума
s=0.01;  % дисперсия
I=zeros(150,150);  % черное изображение
J = imnoise(I,'gaussian',m,s);
imshow(J);          % изображение шума
figure, imhist(J);  % гистограмма изображения шума
```

```
h=imhist(J);
figure,plot(linspace(0,1,length(h)),h);% график гистограммы шума
J2=reshape(J,1,numel(J));  % преобразуем в массив 1 x numel(J)
figure,hist(J2,50);          % гистограмма вектора с 50 корзинами
```



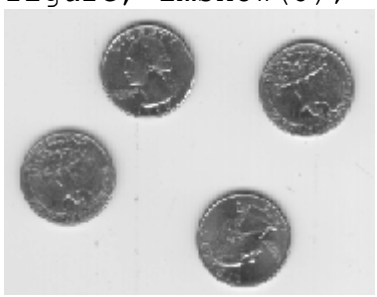


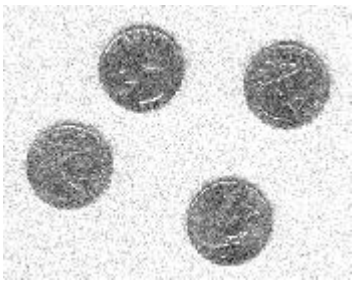




Пример. Наложение шума на реальное изображение

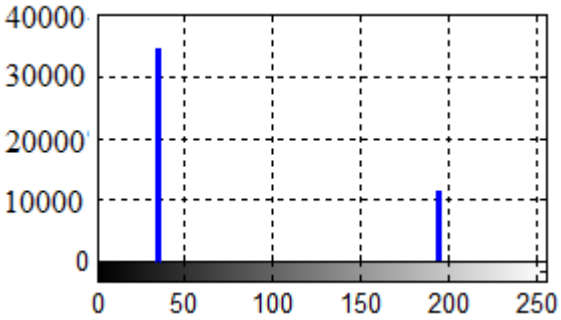
```
I = imread('eight.tif');
imshow(I);
m=0.1; s=0.01;
J = imnoise(I,'gaussian',m,s);
figure, imshow(J);
```

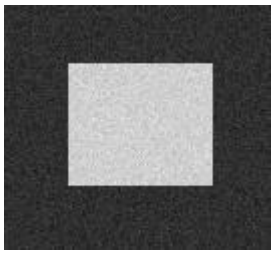


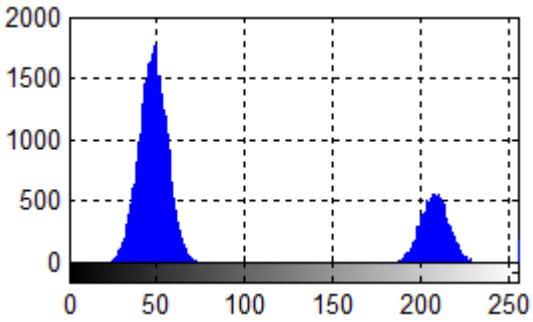


Пример. Наложение гауссова шума на изображение, составленное только из двух яркостных составляющих.

```
f=imread('SquareCircle24.tif');
imshow(f);          % исходное изображение
figure, imhist(f); % гистограмма исходного изображения
axis([0 255 0 40000]); grid on; set(gcf,'Color',[1 1 1]);
% гауссов шум
gn=imnoise(f,'gaussian',0.05,0.001);
figure,imshow(gn);  % зашумленное изображение
figure, imhist(gn); % гистограмма
axis([0 255 0 2000]); grid on;
```





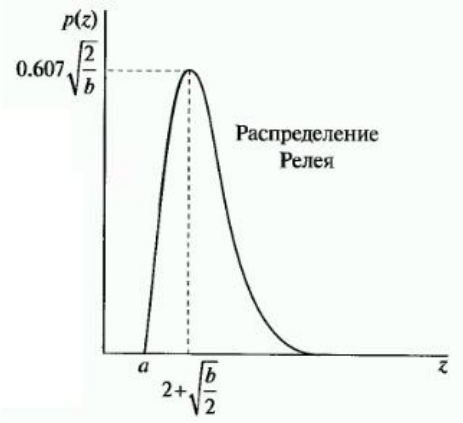


Как видим, исходное изображение содержит пиксели только двух яркостей (I_1=35 – темное, и I_2=190 – светло серое). В зашумленном изображении к этим яркостям пикселей добавляются значения случайной величины с гауссовой функцией плотности распределения вероятности с m=0.05 и σ^2=0.001. В результате пиксели, имевшие одинаковую яркость, принимают немного большие и разбросанные относительно средних яркостей $\tilde{I}_1 = \left(\frac{35}{256} + 0.05\right) \cdot 256 = 49$ и $\tilde{I}_2 = \left(\frac{190}{256} + 0.05\right) \cdot 256 = 203$ значения. Именно этот факт демонстрирует вторая гистограмма.

Шум Релея. Плотность распределения вероятностей шума Релея задается выражением

$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b}, & z \geq a \\ 0, & z < a \end{cases}$$

График плотности распределения вероятностей шума Релея представлен на следующем рисунке.



Обратите внимание на местоположение начала координат и на то, что график имеет асимметричную форму. Математическое ожидание и дисперсия равны

$$\mu = a + \frac{\sqrt{\pi b}}{2} \quad \text{и} \quad \sigma^2 = \frac{b(4-\pi)}{4}$$

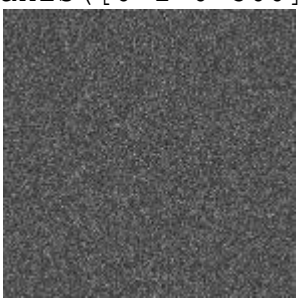
Функция распределения Релея имеет вид

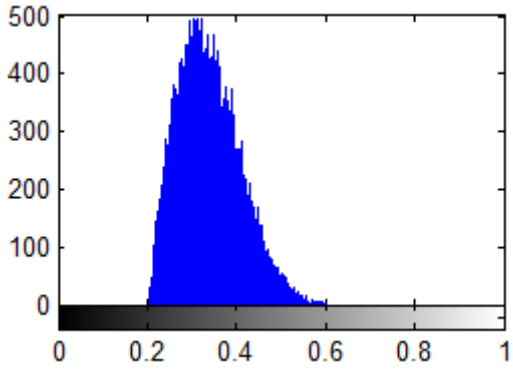
$$F(z) = \begin{cases} 1 - e^{-(z-a)^2/b}, & z \geq a \\ 0, & z < a \end{cases}$$

Чтобы получить выражение случайной величины z с релеевской плотностью вероятности через равномерно распределенную случайную величину w, надо решить уравнение $F(z) = w$, т.е. уравнение $1 - e^{-(z-a)^2/b} = w$. Тогда $z = a + \sqrt{b \ln(1-w)}$.

Пример. Генерирование Рэлеевского шума (exnoise06.m)

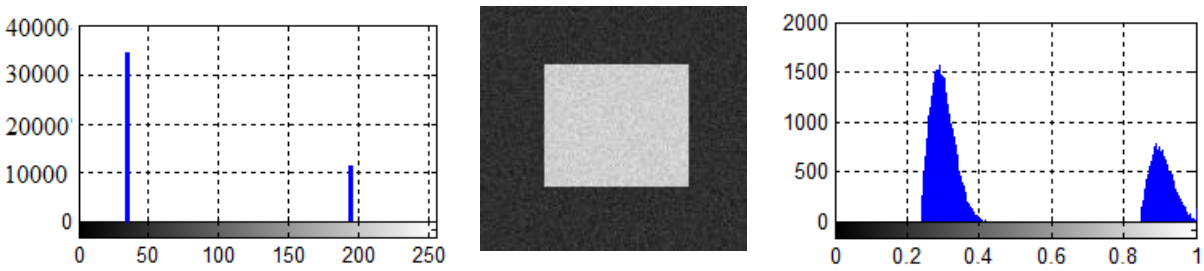
```
close all       % Генерирование Рэлеевского шума
a=0.2;          % Параметры случайной величины
b=0.025;
M=150; N=150;   % Размер изображения
R=a+(-b*log(1-rand(M,N))).^0.5;  % log - натуральный логарифм
imshow(R);
figure, imhist(R); % гистограмма
axis([0 1 0 500]);
```





Пример. Наложения Рэлеевского шума на двухцветное изображение (exnoise06_2.m)

```
close all
a=0.1; b=0.005;
f=imread('SquareCircle24.tif');
figure,imshow(f);
[M,N]=size(f);
R=a+sqrt((-b*log(1-rand(M,N))));
fr=imadd(double(f)/255,R);
figure, imshow(fr);

figure, imhist(double(f)/255);%гистограмма исходного изображения
axis([0 1 0 35000]); grid on;
figure, imhist(fr); % гистограмма
axis([0 1 0 2000]); grid on;
```



На рисунке слева показана гистограмма исходного изображения, в центре – зашумленное изображение и справа – гистограмма зашумленного изображения.

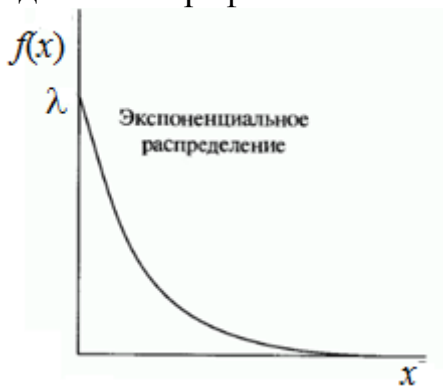
Экспоненциальный шум. Для экспоненциального шума функция плотности распределения вероятности имеет вид (1.6)

$$f(x)=\begin{cases} 0 \ , & x<0 \\ \lambda e^{-\lambda x} \ , & x\geq 0 \end{cases} \quad (\lambda>0)$$

Среднее и дисперсия этого распределения равны (ф. (1.9) и (1.10))

$$\mu=\frac{1}{\lambda}, \ \sigma^2=\frac{1}{\lambda^2}$$

На следующем рисунке представлен график плотности этого распределения



Функция распределения вероятности равна (1.8)

$$F(x)=\begin{cases} 0 \ , & x<0 \\ 1-e^{-\lambda x}, & x\geq 0 \end{cases}$$

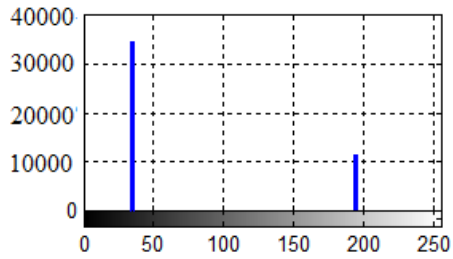
Чтобы получить выражение случайной величины z с экспоненциальной плотностью вероятности через равномерно распределенную случайную величину *w*, надо решить уравнение

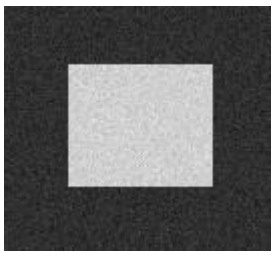
$$1-e^{-\lambda z}=w \quad \text{или} \quad z=-\frac{1}{\lambda}\ln(1-w).$$

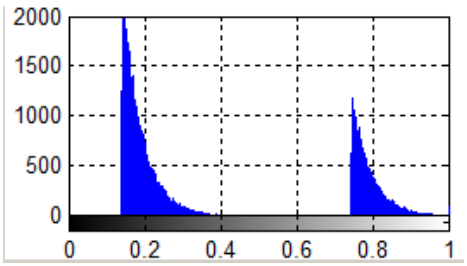
Пример. Наложения экспоненциального шума на двухцветное изображение (exnoise06_3.m)

```
% Генерирование Экспоненциального шума
close all
a=20;
f=imread('SquareCircle24.tif');
figure,imshow(f);
[M,N]=size(f);
R=-1/a*log(1-rand(M,N));
```

```
fr=imadd(double(f)/255,R);
figure, imshow(fr);

figure, imhist(double(f)/255); % гистограмма исходного изображения
axis([0 1 0 35000]); grid on;
figure, imhist(fr); % гистограмма
axis([0 1 0 2000]); grid on;
set(gcf,'Color',[1 1 1]);
```





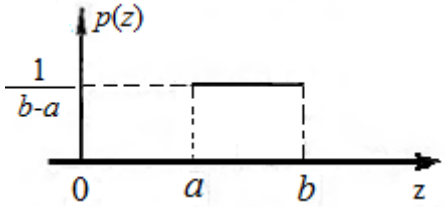


На рисунке слева показана гистограмма исходного изображения, в центре – зашумленное изображение и справа – гистограмма зашумленного изображения.

Равномерный шум. Функция плотности распределения вероятностей равномерного шума задается выражением

$$p(z)=\begin{cases}\dfrac{1}{b-a}, & a\le z\le b;\\ 0\ , & в\ остальных\ случаях\end{cases}$$

Его среднее значение $\mu=\dfrac{a+b}{2}$, а дисперсия $\sigma^2=\dfrac{(b-a)^2}{12}$. На следующем рисунке представлен график плотности такого распределения



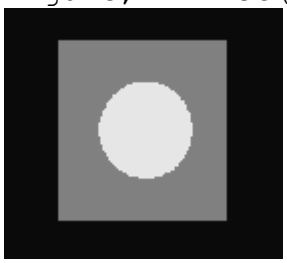
Пример. Наложение равномерного шума.

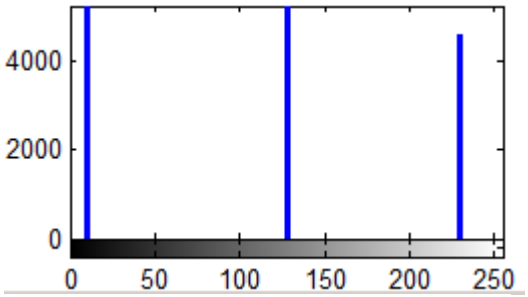
```
% функция генерирование равномерного шума размером M x N
% со значениями из интервала [a b] во всех точах изображения
function R=noise_uniform(M,N,a,b)
R=a+(b-a)*rand(M,N);
```

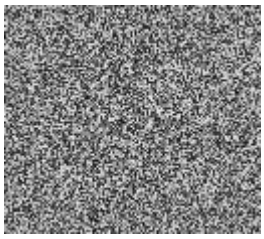
Пример наложения равномерного шума (exnoise15.m)

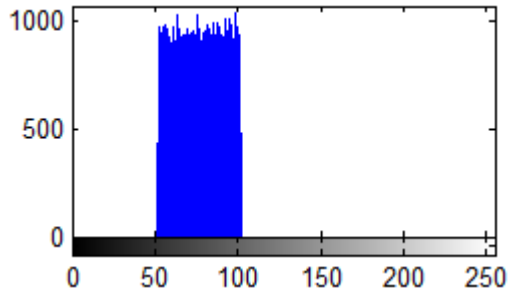
```
% генерирование равномерного шума
close all;
a=0.2; b=0.4;
f=imread('SquareCircle2.tif'); % читаем изображение
[M,N]=size(f);
figure, imshow(f);
% генерируем шум
ru=noise_uniform(M,N,a,b); % равномерный шум во всех точках
figure, imshow(im2uint8(ru));
% портим изображение равномерным шумом
```

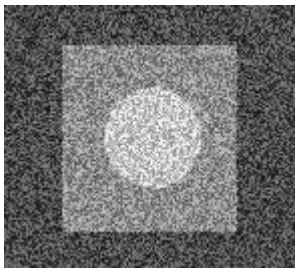
```
fn=imadd(im2double(f),ru);  % наложение шума на изображение
mx=max(max(fn));    % максимальное значение яркости
gn=im2uint8(fn/mx);    % нормирование результирующего изображения
figure, imshow(gn);      % нормирование яркости
%-------------------------------------------------
figure, imhist(f); % гистограмма исходного изображения
figure, imhist(im2uint8(ru)); % гистограмма шума
figure, imhist(im2uint8(gn)); % гист. зашумленного изображения
```

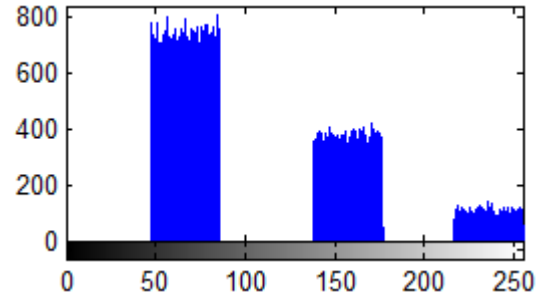










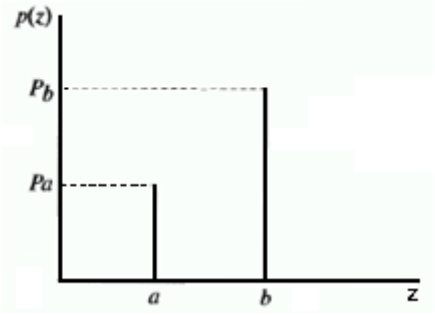


Импульсный шум. Для импульсного (биполярного) шума процесс добавления шума состоит в том, что значение яркости каждой точки изображения с вероятностью $P_s = P_a + P_b \le 1$ заменяется на значение шума. При этом яркость пикселей определяется двумя положительными числами P_a и P_b. Яркость любого пикселя с вероятностью P_a заменяется на значение a, с вероятностью P_b – на значение b и с вероятностью $1 - P_a - P_b$ остается неизменной. Плотность распределения вероятности может быть записана с использованием дельта функции Дирака в виде

$$p(z) = P_a\,\delta(z-a) + P_b\,\delta(z-b)$$

Если $b>a$, то пиксель с яркостью b выглядит как светлая точка на изображении. Пиксель с яркостью a выглядит, наоборот, как темная точка. Если одно из значений вероятности P_a или P_b равно нулю, то импульсный шум называется униполярным. Если ни одна из вероятностей не равна нулю, импульсный шум походит на крупицы соли и перца, рассыпанные по изображению. Поэтому, такой шум часто называют шумом типа «соль и перец».

При оцифровке изображения обычно происходит ограничение значений яркости. Поэтому обычно полагают, что значения a и b равны минимальному и максимальному значениям, которые в принципе могут присутствовать в оцифрованном изображении. Для 8 – битовых изображений это означает, что a=0 (черное, перец), b=255 (белое, соль). График распределения вероятности значений импульсного шума представлен на следующем рисунке.



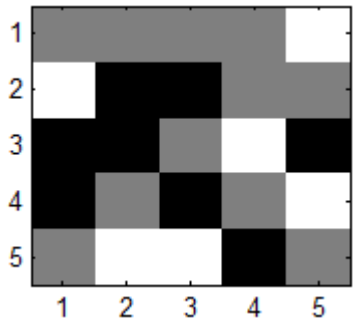
Шум типа «соль и перец» возникает в устройствах с ошибочной коммутацией. Для наложения такого шума на изображение f используется функция

```
g=imnoise(f,'salt & papper',d),
```

где d – плотность шума (процент изображения, подверженного этому шуму). При этом приблизительно $d \cdot \text{numel}(f)$ пикселей будет испорчено. По умолчанию d=0.05. При этом пикселей типа «соль» и типа «перец» образуется примерно одинаковое количество.

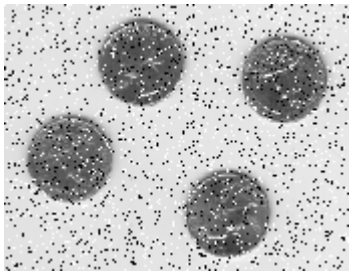
Для пояснения работы функции `imnoise` в этом случае построим изображение шума 5 х 5 пикселей.

```
I= I=ones(5,5)*0.5;  % однотонное изображение (серые пиксели)
J = imnoise(I,'salt & pepper',0.5)  % 0.5 – плотность шума
J3=cat(3,J,J,J) % 3-х мерный массив монохромного изображения
imagesc(J3);  % imagesc рассматривает 3-х мерный массив как RGB
I=ones(150,150)*0.5;
J = imnoise(I,'salt & pepper',0.66);
imshow(J);  % след. рисунок справа
```



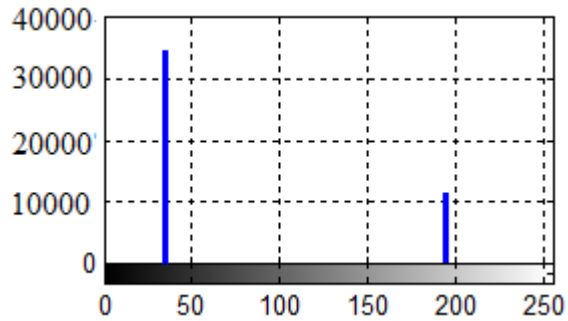
Пример. Наложение шума «соль и перец» на реальное изображение

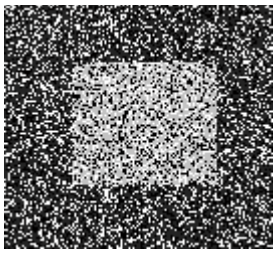
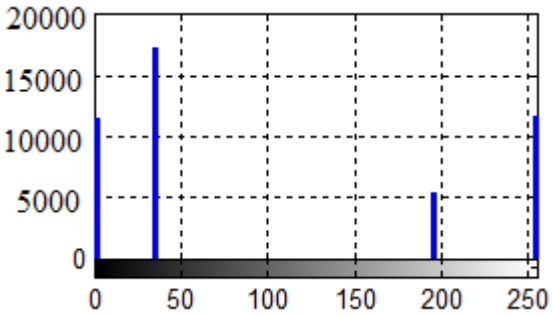
```
I = imread('eight.tif');
imshow(I);
J = imnoise(I,'salt & pepper',0.1);
figure, imshow(J);
```

Пример. Наложение шума «соль и перец» на изображение, составленное из двух яркостных составляющих.

```
f=imread('SquareCircle24.tif');
imshow(f);          % исходное изображение
figure, imhist(f); % гистограмма исходного изображения
axis([0 255 0 40000]); grid on;
set(gcf,'Color',[1 1 1]);
% шум 'соль и перец'
gsp=imnoise(f,'salt & pepper',0.5);
figure,imshow(gsp);  % зашумленное изображение
figure, imhist(gsp); % гистограмма
axis([0 255 0 20000]); set(gcf,'Color',[1 1 1]); grid on;
```

Как видим, исходное изображение содержит пиксели только двух яркостей (I_1=35 – темное, и I_2=190 – светло серое). В зашумленном изображении число пикселей с такими яркостями уменьшилось в двое. Но теперь половина пикселей изображения имеет либо нулевую яркость, либо максимальную – 255. Именно этот факт демонстрирует вторая гистограмма.

Для проверки количества заданных значений яркости пикселей в зашумленном изображении выполним команды

```
ng35=numel(find(gsp==35))  % количество пикселей яркостью 35
ng35 = 16290
ng190=numel(find(gsp==190)) % количество пикселей яркостью 190
```

```
ng190 = 8008
(ng35+ng190)/numel(gsp)  % относительное содержание
ans = 0.5002
```
Как видим, теперь только половина пикселей имеют исходные яркости. Другая половина пикселей имеет нулевую или максимальную яркость.

Иногда нужно сгенерировать шум «только соль» или «только перец» или шум «соль и перец» с разными вероятностями. Тогда функции IPT `imnoise(f,'salt & pepper',d)` недостаточно.

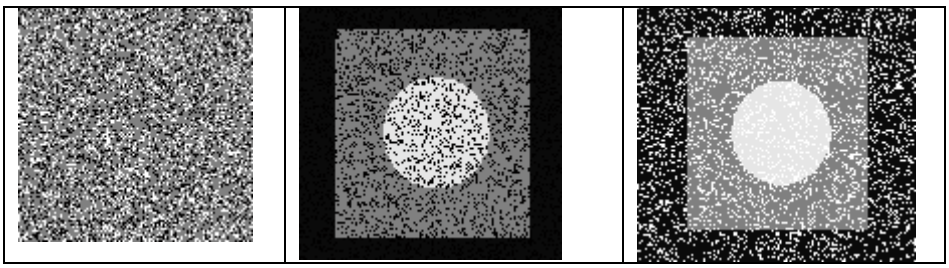
Пример. Функция, которая генерирует изображение шума «соль и перец» с различными вероятностями для соли и перца.

```
% генерирование шума соль и перец
% pb вероятности черного и pa белого шума (pa+pb<=1)
% Если pb=0, то только перец; если pa=0, то только соль
% M,N  размер изображения шума
function R=noise_salt_pepper(M,N,pa,pb)
R(1:M,1:N)=0.5;     % 0.5 признак отсутствия шума в точке
X=rand(M,N);
c=find(X<=pa);
R(c)=0;
c=find(X>pa & X<=pa+pb);
R(c)=1;
```

Пример. Построение изображения шума «соль и перец» и использование функции `noise_salt_pepper( )` для наложения такого шума на изображение (exnoise12.m)

```
% генерирование шума соль и перец
close all;
pa=0.2; pb=0.2;   % вероятности черного и белого шума (pa+pb<=1)
M=150; N=150;    % размер изображения шума
R=noise_salt_pepper(M,N,pa,pb);
imshow(R);          % изображение шума
f=imread('SquareCircle2.tif'); % читаем изображение
[M,N]=size(f);
% портим изображение шумом "соль и перец"
rp=noise_salt_pepper(M,N,0.2,0); % только перец
c=find(rp==0);
gp=f;
gp(c)=0;
figure, imshow(gp);
rs=noise_salt_pepper(M,N,0,0.2); % только соль
c=find(rs==1);
gs=f;
gs(c)=255;
figure, imshow(gs);
```

На следующем рисунке слева показано изображение шума, в центре – изображение, испорченное шумом «только перец», справа – изображение, испорченное шумом «только соль».



Рассмотренные распределения позволяют моделировать искажения, связанные с широким диапазоном встречающихся на практике шумов. Так например, гауссов шум возникает на изображении в результате шума в электронных цепях приемника. Шум Релея возникает на снимках, снятых с большого расстояния. Экспоненциальный шум возникает на изображениях, получаемых с использованием лазеров. Импульсный шум возникает, когда в процессе получения изображения имеют место быстрые переходные процессы.

Периодический шум. Электрические помехи обычно приводят к появлению на изображении периодического шума. Нашей моделью периодического шума будет «двумерная синусоида», задаваемая формулой

$$r(x,y)=A\sin\left[\frac{2\pi u_0}{M}(x+B_x)+\frac{2\pi v_0}{N}(y+B_y)\right]$$

где A – это амплитуда, u_0 и v_0 определяют частоты по горизонтальной и вертикальной осям, B_x и B_y – сдвиги фаз.

Покажем, что дискретным ПФ $R(u,v)$ размера M x N от $r(x,y)$ является пара сопряженных импульсов, находящихся в точках (u_0,v_0) и $(-u_0,-v_0)$. Действительно

$$R(u,v)=\frac{1}{M\cdot N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}r(x,y)e^{-i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}=$$

$$=\frac{A}{2iM\cdot N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left(e^{2\pi i\left(\frac{u_0(x+B_x)}{M}+\frac{v_0(y+B_y)}{N}\right)}-e^{-2\pi i\left(\frac{u_0(x+B_x)}{M}+\frac{v_0(y+B_y)}{N}\right)}\right)e^{-i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}=$$

$$=\frac{A}{2iM\cdot N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left(e^{2\pi i\left(\frac{(u_0-u)x}{M}+\frac{u_0B_x}{M}+\frac{(v_0-v)y}{N}+\frac{v_0B_y}{N}\right)}-e^{-2\pi i\left(\frac{(u_0+u)x}{M}+\frac{u_0B_x}{M}+\frac{(v_0+v)y}{N}+\frac{v_0B_y}{N}\right)}\right)=$$

$$=\frac{A}{2iM}\sum_{x=0}^{M-1}e^{2\pi i\left(\frac{(u_0-u)x}{M}+\frac{u_0B_x}{M}+\frac{v_0B_y}{N}\right)}\frac{1}{N}\sum_{y=0}^{N-1}\left(e^{2\pi i\left(\frac{(v_0-v)y}{N}\right)}\right)-$$

$$-\frac{A}{2iM}\sum_{x=0}^{M-1}e^{-2\pi i\left(\frac{(u_0+u)x}{M}+\frac{u_0B_x}{M}+\frac{v_0B_y}{N}\right)}\frac{1}{N}\sum_{y=0}^{N-1}\left(e^{-2\pi i\left(\frac{(v_0+v)y}{N}\right)}\right)$$

Учитывая условия ортогональности (см. (4.1.10))

$$\sum_{x=0}^{M-1} e^{i\frac{2\pi k x}{M}} e^{-i\frac{2\pi u x}{M}} = \sum_{x=0}^{M-1} e^{i\frac{2\pi}{M}\cdot(k-u)x} = M\cdot\delta_{ku}, \quad \text{где } \delta_{ku} = \begin{cases} 1, k = u \\ 0, k \neq u \end{cases}$$

имеем

$$R(u,v) = \frac{A}{2iM}\left(\sum_{x=0}^{M-1} e^{2\pi i\left(\frac{(u_0-u)x}{M} + \frac{u_0 B_x}{M} + \frac{v_0 B_y}{N}\right)} \delta_{v_0 v} - \sum_{x=0}^{M-1} e^{-2\pi i\left(\frac{(u_0+u)x}{M} + \frac{u_0 B_x}{M} + \frac{v_0 B_y}{N}\right)} \delta_{-v_0 v} \right) =$$

$$= \frac{A}{2i}\left(\delta_{v_0 v} e^{2\pi i\left(\frac{u_0 B_x}{M} + \frac{v_0 B_y}{N}\right)} \frac{1}{M}\sum_{x=0}^{M-1} e^{2\pi i\left(\frac{(u_0-u)x}{M}\right)} - \delta_{-v_0 v} e^{-2\pi i\left(\frac{u_0 B_x}{M} + \frac{v_0 B_y}{N}\right)} \frac{1}{M}\sum_{x=0}^{M-1} e^{-2\pi i\left(\frac{(u_0+u)x}{M}\right)} \right) =$$

$$= \frac{A}{2i}\left(\delta_{v_0 v} e^{2\pi i\left(\frac{u_0 B_x}{M} + \frac{v_0 B_y}{N}\right)} \delta_{u_0 u} - \delta_{-v_0 v} e^{-2\pi i\left(\frac{u_0 B_x}{M} + \frac{v_0 B_y}{N}\right)} \delta_{-u_0 u} \right) =$$

$$= i\frac{A}{2}\left(e^{-2\pi i\left(\frac{u_0 B_x}{M} + \frac{v_0 B_y}{N}\right)} \delta_{-u_0 u}\delta_{-v_0 v} - e^{2\pi i\left(\frac{u_0 B_x}{M} + \frac{v_0 B_y}{N}\right)} \delta_{u_0 u}\delta_{v_0 v} \right)$$

Обозначим $\alpha = i\cdot e^{-2\pi i\left(\frac{u_0 B_x}{M} + \frac{v_0 B_y}{N}\right)}$, тогда $\overline{\alpha} = -i\cdot e^{2\pi i\left(\frac{u_0 B_x}{M} + \frac{v_0 B_y}{N}\right)}$ и мы получаем

$$R(u,v) = \frac{A}{2}\left(\alpha\cdot\delta_{-u_0 u}\delta_{-v_0 v} + \overline{\alpha}\cdot\delta_{u_0 u}\delta_{v_0 v}\right)$$

В результате ДПФ синусоиды имеет вид пары комплексно сопряженных импульсов, которые находятся в точках $(-u_0,-v_0)$ и (u_0,v_0). По – другому, ПФ синусоиды представляет пару сопряженных импульсов, расположенных в центрально – симметричных точках частотной области. Если амплитуды синусоидальных волн в пространственной области достаточно велики, то в спектре изображения будут видны пары импульсов – по одной паре для каждой синусоидальной волны в исходном изображении. Проанализировав фурье – спектр изображения, например, визуально и, обнаружив пики в частотной области, можно оценить параметры периодического шума. Идеальным для обнаружения периодического шума является ПФ серой области, на котором могут быть хорошо заметны пики, соответствующие шуму. Если же доступно только существующее изображение, то можно рассмотреть небольшие участки изображения примерно постоянной яркости.

Пример. В этом примере мы создадим функцию, которая генерирует произвольное количество синусоидальных импульсов (каждый со своей амплитудой, частотой и сдвигом фазы), вычисляя изображения периодического шума $r(x,y)$ размера M x N в виде суммы синусоид. Функция также возвращает преобразование Фурье R(u,v) и спектр S(u,v) созданного шума.

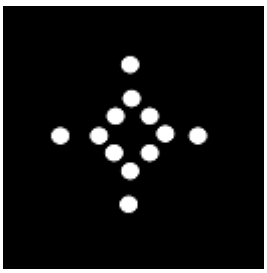
Функция `periodicNoise(...)` генерирует периодический шум, который состоит из суммы нескольких синусоид с разными частотами (u_i,v_i), разными амплитудами A_i и сдвигами фаз (Bx_i,By_i). Функция имеет следующие аргументы: параметр F является K x 2 матрицей, содержащей двумерные

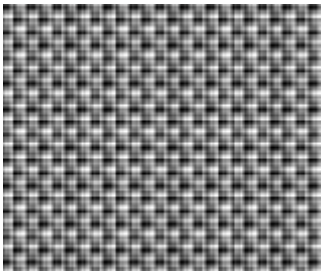
вектора (u_i,v_i), обозначающие положение импульсов в частотной области (относительно центра частотного прямоугольника $(M/2+1, N/2+1)$. A – вектор 1 x K, содержащий амплитуды; B – K x 2 матрица, содержащая сдвиги фаз синусоид.

```
% генерирование периодического шума r - сумма синусоид
% его преобразования фурье R и спектра S
% F - матрица K x 2 частот (отн. центра частотного многоугольн.)
% A - вектор 1 x K амплитуд
% B - матрица K x 2 сдвигов фаз (по умолчанию - нули)
% M и N должны быть четными
function [ r,R,S ] = periodicNoise( M,N,F,A,B)
[K,n]=size(F);
if nargin==3
    A(1:K)=1;
    B(1:K,1:2)=0;
elseif nargin==4
    B(1:K,1:2)=0;
end
M=2*floor(M/2); N=2*floor(N/2);  % делаю четными
R=zeros(M,N);
for j=1:K
    u1=M/2+1+F(j,1);
    v1=N/2+1+F(j,2);
    R(u1,v1)=i*(A(j)/2)*exp(i*2*pi*F(j,1)*B(j,1)/M);
    u2=M/2+1-F(j,1);
    v2=N/2+1-F(j,2);
    R(u2,v2)=-i*(A(j)/2)*exp(i*2*pi*F(j,2)*B(j,2)/N);
end
S=abs(R);
r=real(ifft2(ifftshift(R)));
```

Вот сценарий ее использования (exPeridicNoise01.m)

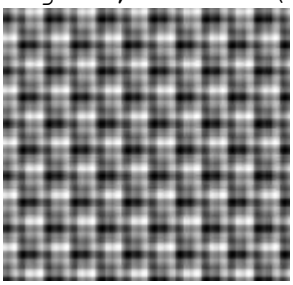
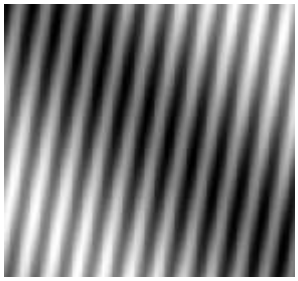
```
% пример генерирования периодического шума
close all;
F=[0 64;0 128; 32 32; 64 0; 128 0; -32 32];
[r,R,S]=periodicNoise(512,512,F);
imshow(S,[]);
figure, imshow(r,[]);
```



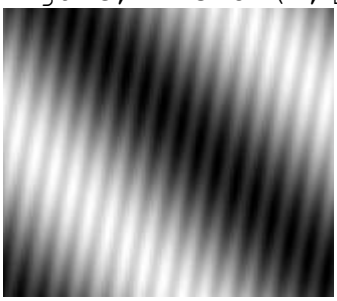


```
F=[0 32;0 64; 16 16; 32 0; 64 0; -16 16];
[r,R,S]=periodicNoise(512,512,F);
% figure,imshow(S,[]);
figure, imshow(r,[]);  % следующий слева

F=[6 32; -2 2];
```

```
[r,R,S]=periodicNoise(512,512,F);
%figure,imshow(S,[]);
figure, imshow(r,[]);  % следующий справа
```

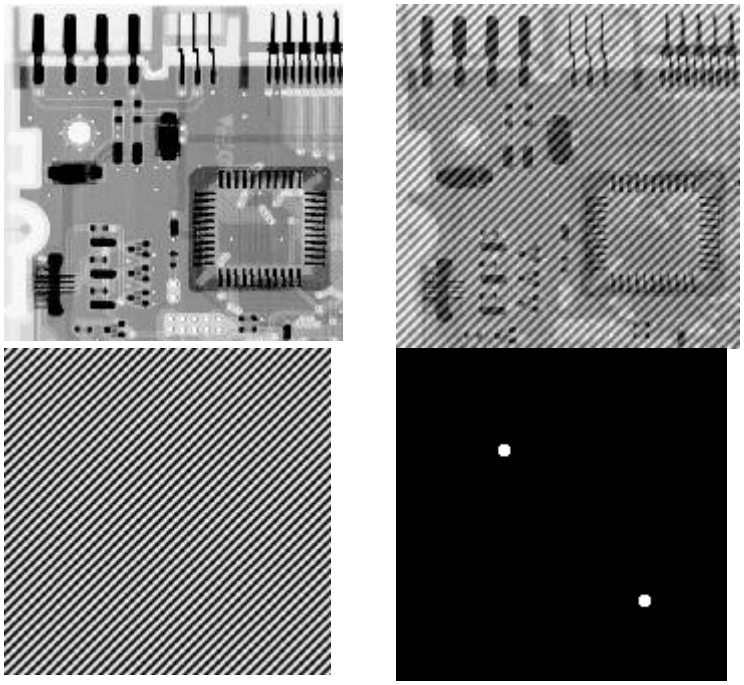
```
F=[6 32; -2 2];
A=[1 5];
[r,R,S]=periodicNoise(512,512,F,A);
%figure,imshow(S,[]);
figure, imshow(r,[]);
```



Вверху слева приведен спектр импульсов (белые точки увеличены для улучшения видимости) и справа соответствующий им шум. Потом приведены изображения различных шумов

Пример. Наложения периодического шума на изображение (exPeridicNoise04.m)

```
% пример наложения периодического шума на изображение
close all;
f=imread('cktboard_200dpi.tif'); % читаем изображение
figure, imshow(f,[]);   %1
[M,N]=size(f);

F=[32 32];                      % частоты импульсных всплесков
[r,R,S]=periodicNoise(M,N,F);   % создаем периодический шум r
figure,imshow(S,[]);            % спектр шума
figure, imshow(r,[]);           % изображение шума

g=imadd(im2double(f),r*100000); % наложение шума с масштабом
figure, imshow(g,[]);           % зашумленное изображение

G=fftshift(fft2(g));          % центрированное ПФ изображения
SG=log(1+1000*G/max(max(G))); % логарифмическое преобразование
figure, imshow(real(SG),[]);  % улучшенное изображение
```



На рисунке вверху слева показано исходное изображение, внизу слева – изображение шума, внизу справа – спектр шума, вверху справа – зашумленное изображение. Для наглядности размер белых точек на спектре шума увеличен.

6.3 Оценивание параметров шума.

В случае пространственного шума параметры плотности распределения вероятности часто определяют с помощью анализа простых тестовых изображений. Задача заключается в оценивании среднего значения и дисперсии по тестовым образцам, которые можно затем использовать для определения параметров уравнений.

Пусть z_i – дискретная случайная величина, значениями которой являются уровни яркости изображения. Обозначим через $p(z_i)$, i=1,2,...,L-1 соответствующую нормированную гистограмму, где L – число возможных значений яркости. Тогда число $p(z_i)$ приближает вероятность появления величины яркости z_i на изображении.

Еще раз. *Нормированную гистограмму можно рассматривать как приближение плотности распределения для яркости*. Тогда среднее значение яркости определится по формуле

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

Дисперсия случайной величины z (яркости) определится по формуле

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

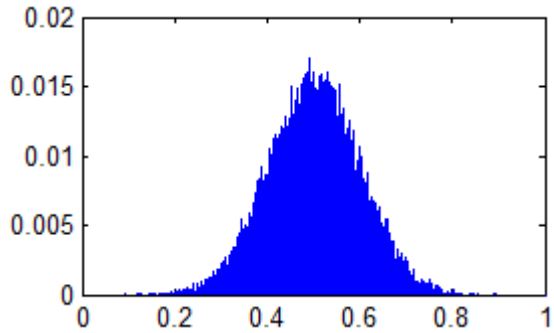
Пусть, например, случайная величина z принимает три значения z=[1 2 3] с вероятностями p=[1/3 1/3 1/3]. Тогда вычислить среднее и дисперсию этой случайной величины можно командами

```
z=[1 2 3];
p=[1/3 1/3 1/3];
m=sum(z.*p)
```

```
m =     2
d=sum((z-m).^2.*p)
d =    0.6667
```

Пример. Построение изображения гауссова шума и определение его параметров по формулам (1) и (2) (ex37.m)

```
close all
m=0.5;   % средняя яркость изображения шума
s=0.01;  % дисперсия
I=zeros(150,150);  % черное изображение
J = imnoise(I,'gaussian',m,s);
imshow(J);          % изображение шума
figure, imhist(J);  % гистограмма изображения шума

% определение параметров изображения шума J
[c,z]=imhist(J);  % c - вектор количеств, z - вектор значений
figure,stem(z,c,'.', 'MarkerSize',1); % строим ту же гистограмму
% нормируем вектор количеств. Он приближенно равен вероятностям
p=c/numel(J);
figure,stem(z,p,'.','MarkerSize',1); %норимированная гистограмма
% здесь p приближают вероятности
m=sum(z.*p) %среднее значение сл.величины z (яркости от 0 до 1)
d=sum((z-m).^2.*p)   % дисперсия сл. величины z
m =    0.5007
d =    0.0100
```



Как видим, вычисленные значения *m* и *d* почти совпали с исходными. А нормированная гистограмма, может приближенно рассматриваться как приближение плотности распределения для яркости.

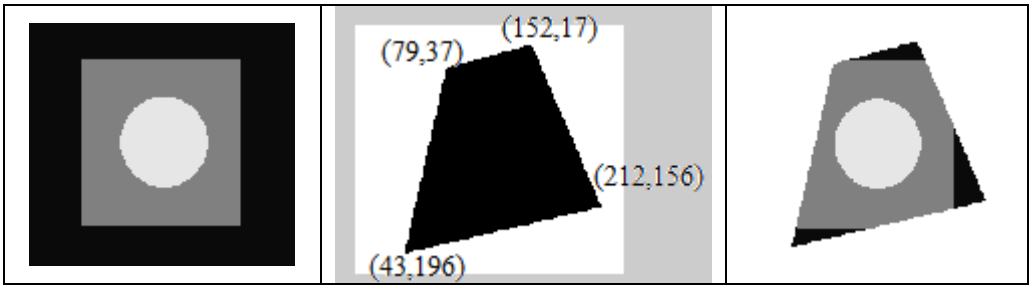
Часто необходимо оцен7ить параметры шума непосредственно по зашумленному изображению. В этом случае следует выбрать на изображении область, по возможности лишенную характерных черт, чтобы распределение яркостей на ней было произведено главным образом шумом. Для выбора такой интересующей области (ROI, Region Of Interest) можно использовать функцию `roipoly`, которая строит многоугольную область. Ее синтаксис имеет вид

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

где `f` – зашумленное изображение, `c` и `r` – векторы горизонтальных и вертикальных координат последовательных вершин многоугольника. Функция возвращает двоичный массив того же размера, что и f, с единицами внутри интересующей области и с нулями вне ее. Массив B затем используется в качестве маски для выделения пикселей выбранной области.

Пример использования функции `roipoly()` (сценарий ex38.m)

```
close all
f=imread('SquareCircle2.tif'); % читаем изображение
imshow(f);
c = [152 212 43 79]; % горизонтальные координаты пикселей вершин
r = [17 156 196 37]; % вертикальные координаты пикселей вершин
B=roipoly(f,c,r);
figure, imshow(imcomplement(B)); % негатив выделенной области

F=ones(size(f))*255;
F(B)=f(B);  % использование бинарной маски для выделения
figure, imshow(uint8(F));
```



На левом рисунке приведено исходное изображение; на среднем – негатив бинарной маски, при этом для пояснения смысла векторов `c` и `r` мы привели координаты вершин. Чтобы увидеть часть изображения, попавшую в пределы выделенной области, предназначены три последние команды сценария.

□

Функцию `roipoly()` можно использовать для интерактивного задания области в виде

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

или

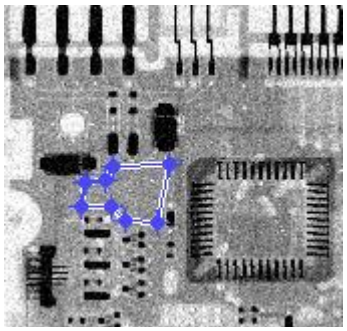
```
[B, c, r]=roipoly(f)
```

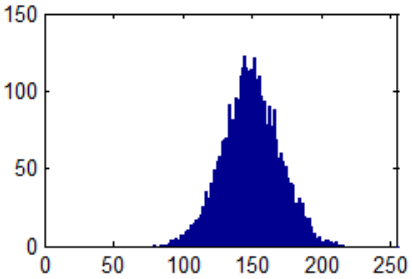
В этом случае изображение f выводится в графическое окно, и пользователю предлагается обозначить интересующую область с помощью мыши. Действуя левой кнопкой мыши следует задать последовательные вершины многоугольника. Когда последнее положение мыши расположено над первой точкой, то курсор принимает форму окружности; потом после щелчка правой кнопкой мыши из выпадающего меню выбираем CreateMask. Вторая форма вызова функции возвращает кроме бинарной маски *B* также вектора горизонтальных `c` и вертикальных `r` координат вершин выделенного многоугольника.

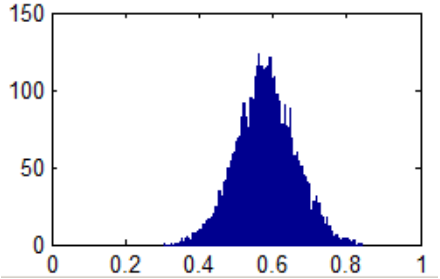
Пример. (exnoise04.m) Определение параметров шума путем построения гистограммы выбранной прямоугольной области на изображении

```
close all
f=imread('Noisy_image1.tif');
imshow(f);
% создание бинарной маски изображения
% выделяем мышью одноцветную область, когда последнее положение
```

```
% мыши расположено над первой точкой, то курсор принимает форму
% окружности; потом правой кнопкой мыши и выбрать CreateMask
[B,c,r]=roipoly(f);
imshow(imcomplement(B));     % негатив выделенной области
% построение гистограммы выделенного изображения
p=imhist(f(B));
figure, bar(p,1); axis([0 255 0 150]);
figure,  hist(double(f(B))/255,256);  axis([0 1 0 150]);
% вычисление среднего m и дисперсии v вектора p гистограммы
% рассматриваемого после нормировки как вероятность
p=p/sum(p); % нормировка
p=p(:);     % преобразование в вектор столбец
z=linspace(0,1,length(p));% созд.норм.сл.величины из диап.[0 1]
m=z*p;      % среднее случайной величины z с вероятностями p
z=z-m;      % центрирование
v=(z.^2)*p; % дисперсия
m,v
m =    0.5795
v =    0.0062
```







Среднее значение m и дисперсия выделенной области на исходном изображении, накрываемом маской B, вычислены по приведенным выше формулам (1) и (2).

□

В последнем примере форма гистограммы указывает на то, что шум имеет гауссов тип. Если форма гистограммы указывает на другой тип шума (Релея, экспоненциальный и т.д.), то следует подобрать тип шума, вычислить среднее и дисперсию и затем использовать их для вычисления параметров функции плотности вероятности выбранного шума.

6.4 Методы удаление шумов

Когда искажение изображения обусловлено исключительно наличием шума, то из модели вытекает, что

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

Слагаемое, описывающее шум, неизвестно, поэтому вычесть его из функции $g(x,y)$ невозможно. Но можно применить пространственную фильтрацию. Она обсуждалась в предыдущих частях нашего пособия, как метод улучшения изображений.

6.4.1 Усредняющие фильтры

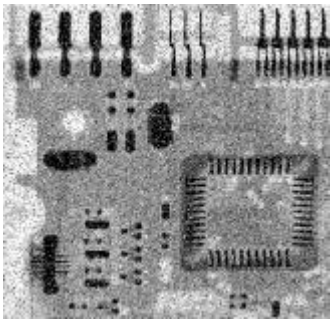
Фильтр, основанный на вычислении среднего арифметического. Пусть S_{xy} обозначает прямоугольную окрестность размера M x N точки (x,y). Значение восстановленного изображения $\hat{f}$ в произвольной точке (x,y) представляет собой среднее арифметическое значений в точках искаженного изображения $g(x,y)$, принадлежащих окрестности S_{xy}.

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

Эта операция может быть реализована в виде свертки с маской, все коэффициенты которой равны $1/M\,N$

Пример. (exnoise08) Удаление шума усредняющим фильтром

```
close all;
f=imread('ChkBoardNoise.tif');
imshow(f);
N=3;
w=ones(N,N)/N^2;
% g1=imfilter(f,w,'replicate');
g1=imfilter(f,w,0);
figure,imshow(g1);
```



Напомним, что функция `imfilter` первым аргументом принимает идентификатор изображения, вторым – матрицу фильтра, дальше – признак расширения границ, могут быть и другие опции. В нашем случае границы исходного изображения расширяются нулем.

Среднегеометрический фильтр. Изображение, восстановленное с использованием среднегеометрического фильтра, задается выражением

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

Тогда

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

и

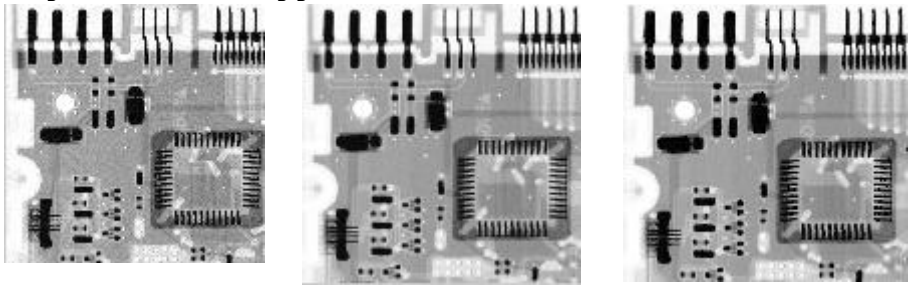
$$\hat{f}(x,y)=\exp\left[\frac{1}{MN}\sum_{(s,t)\in S_{xy}} \ln g(s,t)\right]$$

Для применения среднегеометрического фильтра нужно к логарифму зашумленного изображения применить усредняющий фильтр, а затем применить функцию `exp(...)`.

Очевидно, что применение такого фильтра к изображению зашумленному «перцем» нецелесообразно, т.к. один «нулевой» пиксель делает все произведение нулевым (черным) и на результирующем изображении будет еще больше нулевых/черных пикселей. Обычно применение среднегеометрического фильтра приводит к сглаживанию, сравнимому с применением среднеарифметического фильтра, но при этом теряется меньше деталей.

Пример. Удаление гауссова шума.

```
% удаление гауссова шума
close all;
f=imread('cktboard_gauss.tif');    % изобр. с гауссовым шумом
imshow(f);
% среднеарифметический фильтр
N=3; w=ones(N,N)/N^2;
% g1=imfilter(f,w,'replicate');
g1=imfilter(fg,w);
figure,imshow(g1);
% среднегеометрический фильтр
N=3;
g=im2double(f); % преобразование к double 0 - 1
warning off; % отключить предупреждение log(0)
gg=exp(imfilter(log(g),ones(N,N),'replicate')./(N.^2));
ff=im2uint8(gg); % обратное преобразование к uint8 0 - 255
figure,imshow(gg);
```

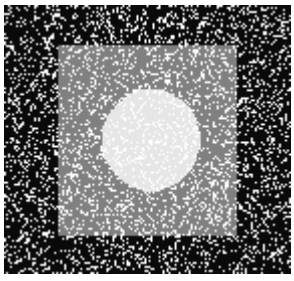


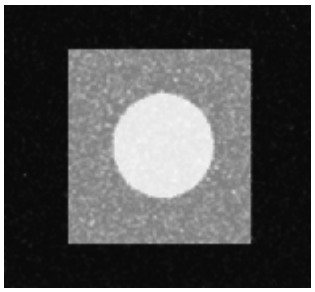
Среднегармонический фильтр. Результат обработки таким фильтром дается выражением

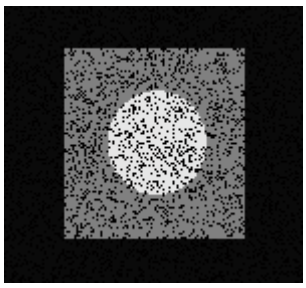
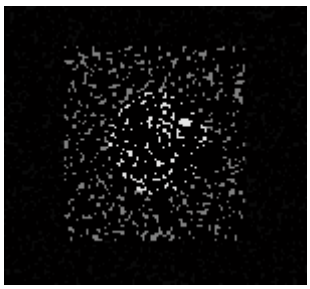
$$\hat{f}(x,y)=\frac{M\,N}{\sum_{(s,t)\in S_{xy}}\frac{1}{g(s,t)}}$$

Пример. Применение среднегармонического фильтра к изображению с шумом «только соль» и «только перец» (exnoise13.m). Здесь используется созданная нами ранее функция `noise_salt_pepper(...)`.

```
% применение среднегармонического фильтра к изображению
% с белым и черным импульсными шумами
close all;
f=imread('SquareCircle2.tif');

[M,N]=size(f);
rs=noise_salt_pepper(M,N,0,0.2); % только соль
c=find(rs==1);
gs=f;
gs(c)=255;
figure, imshow(gs)

n=3;
g=im2double(gs); % преобразование к double 0 - 1
% среднегармонический фильтр
gg=n.^2./imfilter(1./(g+eps),ones(n,n),'replicate');
g1=im2uint8(gg); % обратное преобразование к uint8 0 - 255
figure,imshow(g1);

rp=noise_salt_pepper(M,N,0.2,0); % только перец
c=find(rp==0);
gp=f;
gp(c)=0;
figure, imshow(gp);

g=im2double(gp); % преобразование к double 0 - 1
% среднегармонический фильтр
gg=n.^2./imfilter(1./(g+eps),ones(n,n),'replicate');
g2=im2uint8(gg); % обратное преобразование к uint8 0 - 255
figure,imshow(g2);
```





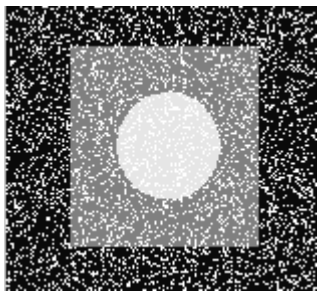
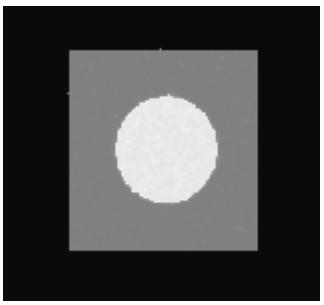
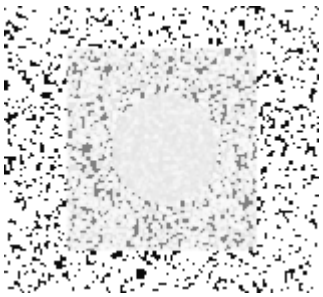
Среднегармонический фильтр хорошо работает в случае наличия только «белого» импульсного шума, но не работает в случае «черного» импульсного шума. Этот фильтр неплохо работает для других типов шумов, таких как гауссов шум.

Фильтр, основанный на вычислении среднего контрагармонического.
Обработка изображения с использованием такого фильтра описывается выражением

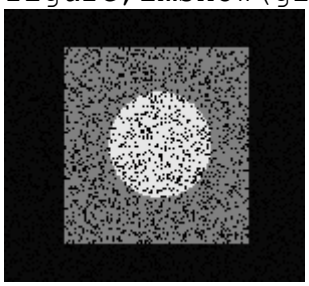
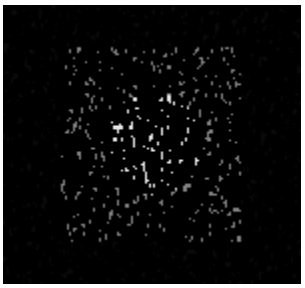
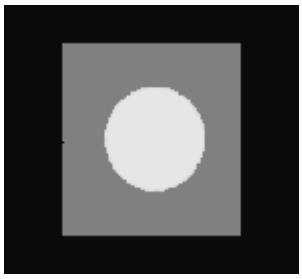
$$\hat{f}(x,y)=\frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

где Q называется порядком фильтра.
Пример. (exnoise14.m) Применение среднего контрагармонического фильтра к изображению с белым и черным импульсными шумами

```
close all;
f=imread('SquareCircle2.tif');

[M,N]=size(f);
rs=noise_salt_pepper(M,N,0,0.2); % только соль
c=find(rs==1);
gs=f;
gs(c)=255;
figure, imshow(gs);  %

g=im2double(gs); % преобразование к double 0 - 1
n=3;
% среднеконтрагармонический фильтр с отрицательным порядком q
q=-5;
if1=imfilter(g.^(q+1),ones(n,n),'replicate');
gg=if1./(imfilter(g.^q,ones(n,n),'replicate')+eps);
g1=im2uint8(gg); % обратное преобразование к uint8 0 - 255
figure,imshow(g1);

% среднеконтргармонический фильтр с положительным порядком q
q=5;
if1=imfilter(g.^(q+1),ones(n,n),'replicate');
gg=if1./(imfilter(g.^q,ones(n,n),'replicate')+eps);
g2=im2uint8(gg); % обратное преобразование к uint8 0 - 255
figure,imshow(g2);
```

```
rp=noise_salt_pepper(M,N,0.2,0); % только перец
c=find(rp==0);
gp=f;
gp(c)=0;
figure, imshow(gp);

g=im2double(gp); % преобразование к double 0 – 1
% среднеконтргармонический фильтр с отрицательным порядком q
q=-5;
if1=imfilter(g.^(q+1),ones(n,n),'replicate');
gg=if1./(imfilter(g.^q,ones(n,n),'replicate')+eps);
g3=im2uint8(gg); % обратное преобразование к uint8 0 – 255
figure,imshow(g3);
% среднеконтргармонический фильтр с положительным порядком q
q=5;
if1=imfilter(g.^(q+1),ones(n,n),'replicate');
gg=if1./(imfilter(g.^q,ones(n,n),'replicate')+eps);
g2=im2uint8(gg); % обратное преобразование к uint8 0 – 255
figure,imshow(g2);
```

При $Q > 0$ фильтр устраняет «черную» составляющую импульсного шума, а при Q<0 – «белую». Обе части шума не могут быть устранены одновременно. Но можно одним фильтром (Q<0) вначале удалить «белую» составляющую шума, а затем к результату применить фильтр с Q>0, который удалит «черную» составляющую.

Заметим, что контрагармонический фильтр при Q=0 сводится к среднеарифметическому фильтру, а при Q=-1 сводится к среднегармоническому фильтру.

6.4.2 Фильтры порядковых статистик

Фильтры, основанные на порядковых статистиках, представляют собой пространственные фильтры, вычисление отклика которых требует предварительного упорядочивания значений пикселей, заключенных внутри обрабатываемой фильтром области изображения. Отклик фильтра в любой точке определяется результатом упорядочивания.

Наиболее употребительные фильтры порождаются в пакете IPT функцией `ordfilt2`, которая строит фильтры порядковых статистик (их также называют ранговыми фильтрами). Отклики этих фильтров основаны на предварительном упорядочении (ранжировании) пикселей изображения из текущей окрестности, после чего центральному пикселю присваивается значение, определенное в результате данного упорядочения. Синтаксис функции `ordfilt2` имеет следующий вид:

```
g = ordfilt2(f, order, domain);
```

Эта функция создает выходное изображение `g`, заменяя каждый элемент `f` на элемент с номером `order` в упорядоченной последовательности ненулевых элементов из окрестности, заданной параметром `domain`. Здесь `domain` — это матрица `m x n`, состоящая из нулей и единиц, которые обозначают позиции пикселей, участвующие в вычислениях. В этом смысле матрица `domain` действует как маска. Пиксели окрестности, которым соответствуют нули в `domain`, не участвуют в вычислениях. Например, чтобы реализовать фильтр минимума размера `m x n`, применяется команда

```
g = ordfilt2(f, 1, ones(m, n));
```

В такой записи `order=1` означает первый элемент в упорядоченной последовательности из `m·n` пикселей, а `ones(m,n)` — это матрица `m x n`, из одних единиц, указывающая на то, что все элементы окрестности участвуют в вычислении отклика фильтра. Аналогично, фильтр максимума реализуется командой

```
g = ordfilt2(f, m*n, ones(m, n));
```

Самым известным фильтром порядковых статистик в цифровой обработке изображений является медианный фильтр, который соответствует среднему элементу маски. Для этого используют функцию `median` в `ordfilt2` следующим образом

```
g = ordfilt2(f, median(1:m*n), ones(m, n));
```

Здесь `median(1:m*n)` — это медиана упорядоченной последовательности чисел 1,2,..., `m·n` (число, занимающий срединное положение в упорядоченной последовательности). Например

```
median([3  5  17  34  21  24 -21])
ans =  17
sort([3  5  17  34  21  24 -21])
ans = -21     3     5    17    21    24    34
```

Число 17 расположено в середине упорядоченной последовательности (при нечетном количестве элементов). Или

```
median([3  5  17 34  21  24])
```

```
ans =  19
sort([3 5 17 34 21 24])
ans =   3     5    17    21    24    34
```
Число 19 является средним из двух чисел 17 и 21, которые расположены в середине упорядоченной последовательности (при четном количестве элементов).

В силу практической важности медианного фильтра в пакете IPT предусмотрена специальная реализация этого фильтра:

```
g = medfilt2(f, [m, n], padopt),
```

где пара `[m,n]` задает окрестность размера `m x n`, по которой вычисляется медиана. Параметр `padopt` задает три возможные опции расширения границ изображения. Она может принимать значение по умолчанию '`zeros`' с нулевым расширением. Второе значение '`symmetric`', расширяет изображение `f` путем его зеркального отражения через границы. Третье значение '`indexed`', при котором f расширяется значением 1, если `f` имеет класс `double` и значением 0 в противном случае. В простейшем виде эта функция вызывается командой

```
g = medfilt2(f);
```

При этом используется окрестность 3 x 3 для вычисления медианы, и входное изображение расширяется нулевыми значениями пикселей.

Медианные фильтры. Самым известным фильтром порядковых статистик в цифровой обработке изображений является медианный фильтр, который соответствует среднему элементу маски. Действие медианного фильтра состоит в замене значения в точке изображения на медиану значений яркости в окрестности этой точки

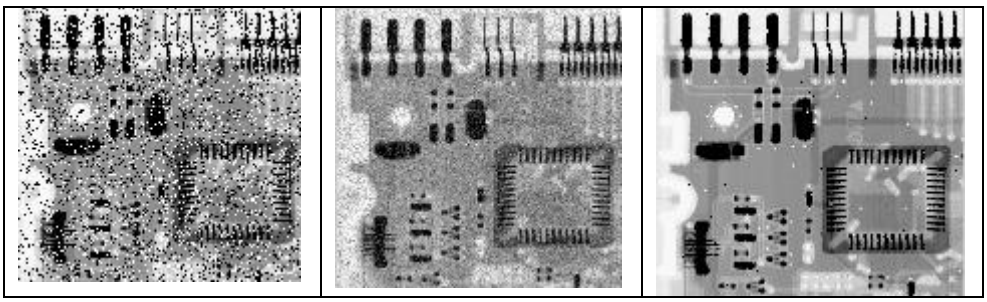
$$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{med}\{g(s,t)\}$$

Медианные фильтры особенно эффективны при наличии биполярного или униполярного импульсного шума.

Пример. (exnoise09.m)

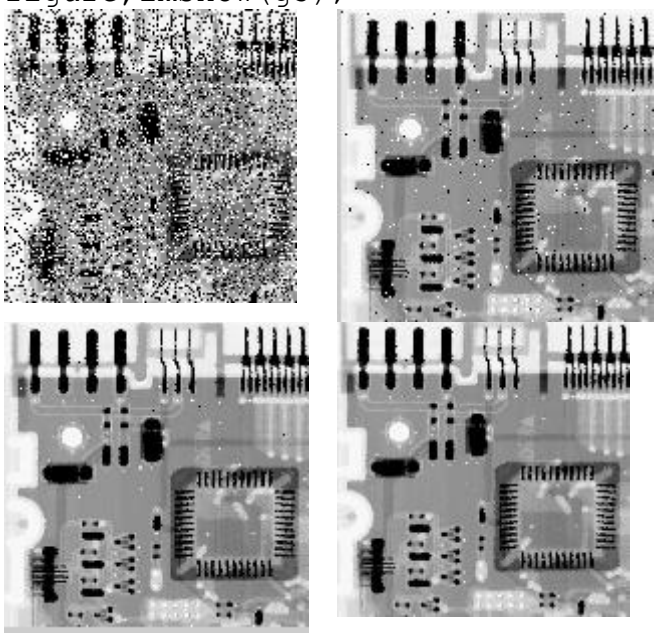
```
f=imread('ChkBoardNoise.tif');
imshow(f);
figure, imhist(f); % гистограмма исходного изображения
% среднеарифметический фильтр
N=3;
g1=imfilter(f,ones(N,N)/N^2,'replicate');
figure,imshow(g1);
% медианный фильтр
g2=medfilt2(f,[N N],'symmetric');
figure,imshow(g2);
```

На следующем рисунке слева представлено изображение с шумом «соль и перец», в середине – восстановление усредняющим арифметическим фильтром и справа – восстановленное с использованием медианного фильтра.



Пример. Трехкратное применение медианного фильтра (exnoise10.m)

```
close all;
f=imread('cktboard_200dpi.tif'); % чистое изображение
imshow(f);
% наложение импульсного шума
fg=imnoise(f,'salt & pepper',0.3);
figure,imshow(fg);  % зашумленное изображение
% применение медианного фильтра
g1=medfilt2(fg,[N N],'symmetric');
figure,imshow(g1);
% применение медианного фильтра повторно
g2=medfilt2(g1,[N N],'symmetric');
figure,imshow(g2);
% применение медианного фильтра третий раз
g3=medfilt2(g2,[N N],'symmetric');
figure,imshow(g3);
```



Представленное вверху слева изображение искажено биполярным импульсным шумом с вероятностями $P_a = P_b = 0.15$. Вверху справа представлен результат обработки с использованием медианного фильтра размером 3 x 3. Улучшение изображения весьма значительно, однако по-прежнему можно видеть остатки шума в некотором количестве точек. Второй проход по изображению медианным фильтром (второй ряд слева) удаляет большинство этих точек, оставляя едва заметную их часть. Последние удаляются при третьем проходе.

Этот пример наглядно демонстрирует силу медианной фильтрации при обработке изображений, содержащих импульсный шум. Однако следует иметь в виду, что многократное применение медианной (и любой другой) фильтрации приводит к постепенному размыванию изображения, поэтому, желательно, чтобы число проходов было как можно меньше.

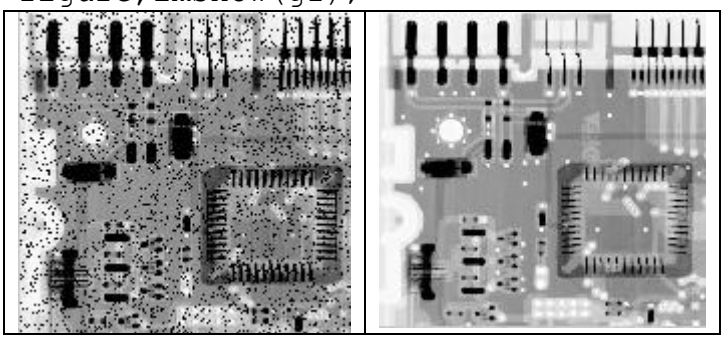
Фильтры, основанные на выборе минимального и максимального значения. Фильтр максимума задается выражением

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

Поскольку униполярный «черный» импульсный шум принимает минимальные значения, применение этого фильтра приводит к уменьшению такого шума, поскольку в процессе фильтрации из окрестности S_{xy} выбирается максимальное значение.

Пример. Применение фильтра максимума

```
f=imread('ckt_pepper_only.tif');
figure,imshow(f);
% фильтр максимума
g1=ordfilt2(f,N*N,ones(N,N),'symmetric');
figure,imshow(g1);
```



Здесь слева изображение с шумом только перец, а справа – восстановленное фильтром максимума.

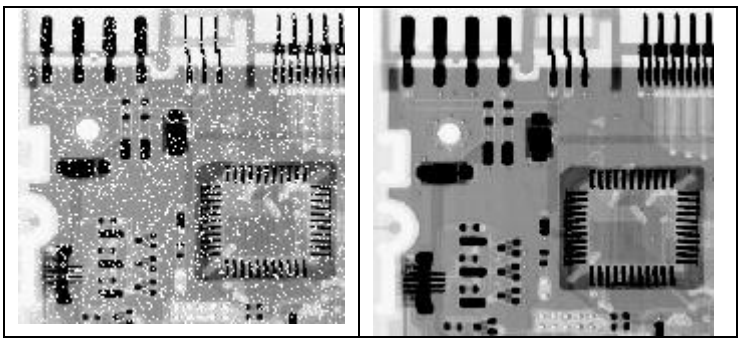
Фильтр минимума задается выражением

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

Применение этого фильтра приводит к уменьшению униполярного «белого» импульсного шума, поскольку в процессе фильтрации из окрестности S_{xy} выбирается минимальной значение.

Пример. Применение фильтра минимума

```
f=imread('ckt_salt_only.tif');
figure,imshow(f);
% фильтр минимума
g2=ordfilt2(f,1,ones(N,N),'symmetric');
figure,imshow(g2);
```



Здесь слева изображение с шумом только соль, а справа – восстановленное фильтром минимум.

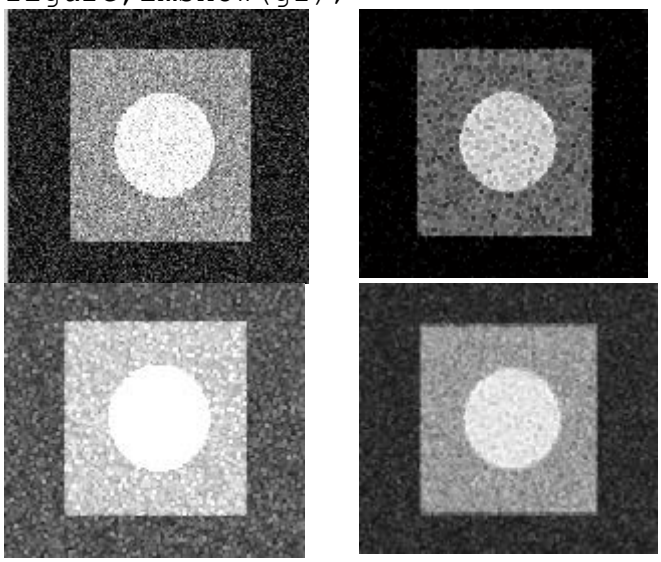
Фильтр – срединная точка. Применение этого фильтра заключается в вычислении среднего между максимальным и минимальным значениями в соответствующей окрестности

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

Этот фильтр лучше всего работает при наличии таких случайно распределенных шумов, как гауссов или равномерный.

Пример. (exnoise11.m) Применение фильтра «срединная точка».

```
% удаление шумов
close all;
f=imread('SquareCircle2.tif');
% imshow(f);
% наложение гауссова шума
fg=imnoise(f,'gaussian',0.1,0.02);
figure,imshow(fg);  % зашумленное изображение
% обработка фильтром срединная точка
N=3;
f1=ordfilt2(fg,1,ones(N,N),'symmetric');
figure,imshow(f1);
f2=ordfilt2(fg,N*N,ones(N,N),'symmetric');
figure,imshow(f2);
g1=imlincomb(0.5,f1,0.5,f2);
figure,imshow(g1);
```



Фильтры усеченного среднего. Предположим, что мы удалили $d/2$ наименьших и $d/2$ наибольших значений яркости из окрестности S_{xy}. Пусть $g_r(s,t)$ представляет собой оставшиеся элементы изображения этой окрестности. Фильтр, действие которого заключается в усреднении оставшихся значений, называется фильтром усеченного среднего

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

Здесь m x n определяют размер окрестности S_{xy} и значение d может изменяться в диапазоне от 0 до $m \cdot n - 1$. В случае d=0, фильтр усеченного среднего сводится к среднеарифметическому фильтру, а в случае $d/2=(mn-1)/2$ превращается в медианный фильтр. Использование фильтра усеченного среднего с другими значениями d полезно в случаях, когда на изображении присутствует несколько видов шума одновременно, например, импульсный и гауссов шумы.

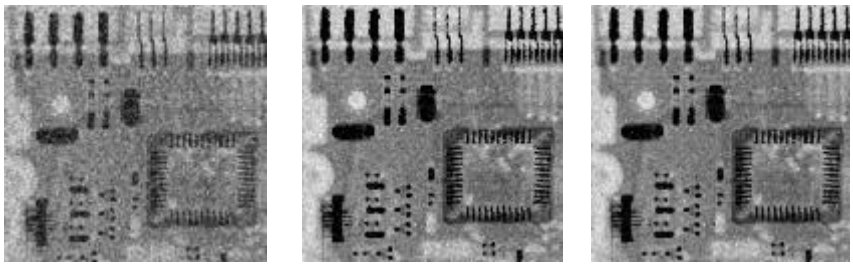
Код функции, реализующей фильтрацию усеченного среднего

```
% фильтр усеченного среднего
% удалить d наименьших и d наибольших значений яркости
% изображения f типа uint8
% m,n - размер фильтра
function g=trimmean(f,m,n,d)
if (d<1) | (d>m*n-1)
    error('bad parameter d')
end
g=im2double(f);
ff=imfilter(g,ones(m,n),'symmetric');  % суммируем все пиксели
окрестности
for k=1:d     % вычитаем d наименьших значений
    ff=imsubtract(ff,ordfilt2(g,k,ones(m,n),'symmetric'));
end
for k=(m*n-d+1):(m*n)    % вычитаем d наибольших значений
    ff=imsubtract(ff,ordfilt2(g,k,ones(m,n),'symmetric'));
end
ff=ff/(m*n-2*d);
g=im2uint8(ff);
```

Пример. Приведен код сценария, в котором мы портим изображение равномерным шумом и шумом 'соль и перец', а затем сравниваем результаты фильтрации среднеарифметическим фильтром, медианным фильтром и фильтром усеченного среднего (exnoise17.m).

```
close all;
f=imread('cktboard_200dpi.tif');
[M,N]=size(f);
% генерируем равномерный шум со вредним m=(a+b)/2=0
a=-0.2; b=0.2;
ru=noise_uniform(M,N,a,b); % равномерный шум во всех точках
figure, imshow(im2uint8(ru));
% портим изображение равномерным шумом
```

```
fn=imadd(im2double(f),ru);  % наложение шума на изображение
mx=max(max(fn));    % максимальное значение яркости
gn=im2uint8(fn/mx); % нормирование результирующего изображения
figure, imshow(gn);
figure, imhist(f);  % гист. исх. изображения
figure, imhist(gn); % гист. зашумленного изображения
% добавление шума 'соль и перец'
fesp=imnoise(gn,'salt & pepper',0.3);
figure, imshow(fesp);
% -------------- фильтрация -----------------------
% среднеарифметический фильтр
p=5; % размер фильтра 5 x 5
g1=imfilter(fesp,ones(p,p)/p^2,'replicate');
figure,imshow(g1);
% медианный фильтр
g2=medfilt2(fesp,[p p],'symmetric');
figure,imshow(g2);
% фильтр усеченного среднего
g3=trimmean(fesp,p,p,8);
figure,imshow(g3);
```



На левом рисунке показан результат фильтрации с использованием среднеарифметического фильтра, на среднем – медианного фильтра, на правом – фильтра усеченного среднего.

6.4.3 Подавление периодического шума

Подавление периодического шума выполняется с помощью фильтрации в частотной области.

Как мы видели, периодический шум на спектре изображения проявляется в виде импульсно подобных всплесков. Основной метод борьбы основан на режекторной фильтрации – фильтрации, которая не пропускает сигналы (световые колебания) с частотами некоторого определенного диапазона и пропускает сигналы со всеми другими частотами. Узкополосные режекторные фильтры не пропускают частоты в некоторых окрестностях своих центральных частот.

Передаточная функция узкополосного режекторного фильтра Баттерворта порядка n имеет вид

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

где

$$D_1(u,v)=\sqrt{\left(u-\frac{M}{2}-u_0\right)^2+\left(v-\frac{N}{2}-v_0\right)^2}$$

$$D_2(u,v)=\sqrt{\left(u-\frac{M}{2}+u_0\right)^2+\left(v-\frac{N}{2}+v_0\right)^2}$$

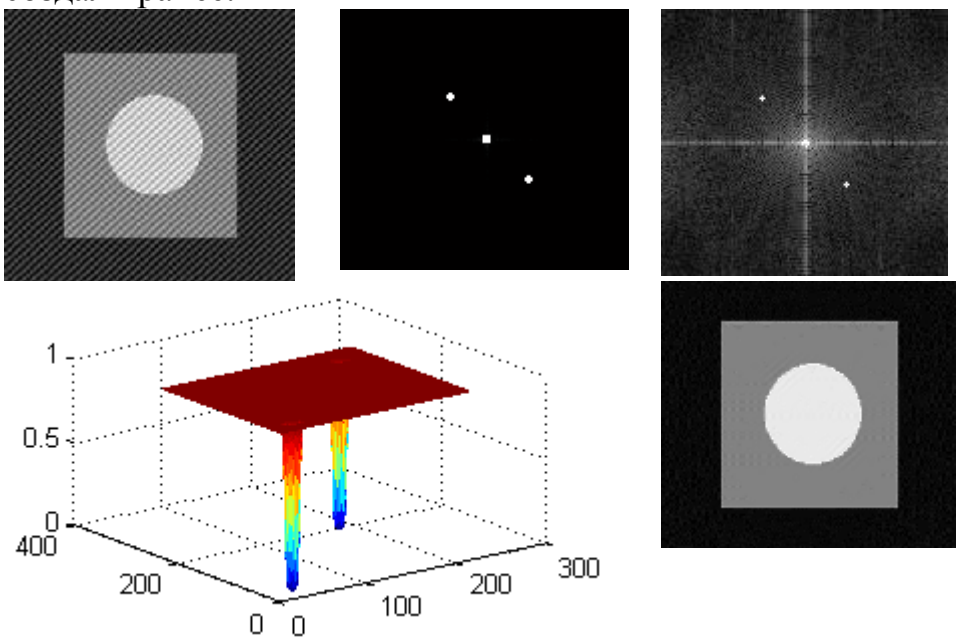
Здесь (u_0,v_0) и $(-u_0,-v_0)$ - центры расположения частотных всплесков «радиуса» D_0. При этом фильтр задан относительно центра частотного прямоугольника и перед использованием в Matlab его следует обработать функцией `fftshift`.

Пример. (exPeridicNoise03.m) В настоящем примере мы читаем файл изображения, создаем шум и накладываем его на изображение, т.е. создаем изображение с периодическим шумом. Затем мы создаем фильтр Баттерворта и применяем его к ПФ зашумленного изображения, т.е. выполняем фильтрацию в частотной области. Затем выполняем ОПФ, восстанавливая изображение. В комментариях сценария указаны номера графических окон, которые создаются соответствующими строками кода.

```
% пример наложения периодического шума на изображение
% и фильтрация такого шума
close all;
f=imread('SquareCircle2even.tif'); % читаем изображение
% f=imread('cktboard_200dpi.tif'); % читаем изображение
figure, imshow(f,[]);   %1
[M,N]=size(f);

F=[32 32];                          % частоты импульсных всплесков
[r,R,S]=periodicNoise(M,N,F);   % создаем периодический шум r
figure,imshow(S,[]);                %2 центрированный спектр шума
figure, imshow(r,[]);               %3 изображение периодического шума

g=imadd(im2double(f),r*10000);  % накладываем периодический шум с
масштабом на изображение
figure, imshow(g,[]);               %4 зашумленное изображение

fg=fft2(g);          % ПФ зашумленного изображения
(нецентрированное)
fgs=fftshift(fg);    % центрирование ПФ
sg=abs(fgs);
figure, imshow(sg,[]);  %5 изображение центрированного спектра
s2=log(1+sg);           % логарифмическое преобразование
figure, imshow(s2,[]);  %6 улучшенное изображение

% видим две яркие точки на диагонали (не в центре); это шум, его
надо отфильтровать
% ---------- Создаем режекторный фильтр Баттерворта --------
D1=@(u,v,u0,v0,M,N)          ((u-M/2-u0).^2+(v-N/2-v0).^2).^(1/2);
D2=@(u,v,u0,v0,M,N)          (((u-M/2+u0).^2+(v-
N/2+v0).^2)).^(1/2);
```

```
DD=@(u,v,u0,v0,M,N,D0,n)
(D0^2./(D1(u,v,u0,v0,M,N).*D2(u,v,u0,v0,M,N))).^n;
HB=@(u,v,u0,v0,M,N,D0,n) 1./(1+DD(u,v,u0,v0,M,N,D0,n)); % функция
центрированного фильтра
[U,V]=meshgrid(1:M,1:N);
HHB=HB(U,V,32,32,M,N,15,8);
figure,mesh(U,V,HHB);  %7 поверхность центр. фильтра Баттерворта
HHBB=fftshift(HHB);    %  децентрирование фильтра
figure,mesh(U,V,HHBB); %8 пов. децентр. фильтра Баттерворта
%-----------------------------------------------------------
% фильтрация в частотной области и ОПФ
g=real(ifft2(fg.*HHBB'));   % fg нецентрированное ПФ изображения,
HHBB - децентрированный фильтр
figure, imshow(g,[]);       %9 восстановленное изображение
```

Заметим, что здесь мы использовали функцию `periodicNoise`, которую создали ранее.



На рисунке вверху слева показано зашумленное изображение, вверху в центре – его центрированный спектр, вверху справа – преобразованный спектр, внизу слева – децентрированный фильтр Баттерворта, внизу справа – результат частотной фильтрации, т.е. восстановленное изображение. Для наглядности точки, представляющие частотные всплески, на изображениях спектров увеличены.

Узкополосные фильтры пропускают, а не подавляют частоты в окрестностях некоторых своих центральных частот. Их передаточные функции получаются по формулам

$$H_{np}(u,v)=1-H_{nr}(u,v)$$

где $H_{np}(u,v)$ - передаточная функция узкополосного (пропускающего) фильтра, соответствующая режекторному фильтру $H_{nr}(u,v)$. Применяя к изображению с периодическим шумом передаточную функцию $H_{np}(u,v)$, мы отфильтруем все

детали изображения, оставив только составляющие шума. Например, в продолжение предыдущего сценария (exPeridicNoise03.m) выполним команды

```
% узкополосная фильтрация - выделение шумовой составляющей
PH=1-HHBB';
pg=real(ifft2(fg.*PH));
figure, imshow(pg,[]);
```

□

Кольцевые режекторные фильтры удаляют или ослабляют частоты в кольцевой зоне вокруг начала координат частотной области.

Передаточная функция идеального кольцевого режекторного фильтра задается выражением

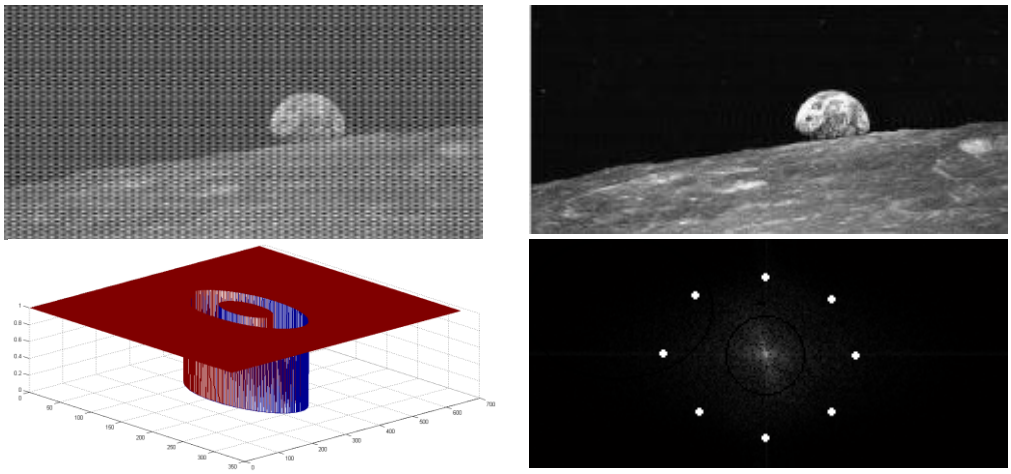
$$H(u,v)=\begin{cases}1, & D(u,v)<D_0-\dfrac{W}{2}\\ 0, & D_0-\dfrac{W}{2}\le D(u,v)<D_0+\dfrac{W}{2}\\ 1, & D(u,v)>D_0+\dfrac{W}{2}\end{cases}$$

где $D(u,v)$ - расстояние, измеряемое от центра частотного прямоугольника, D_0 – радиус окружности, проходящей через середину кольца, W – ширина кольца.

Пример. В примере (exPeridicNoise07.m) мы читаем зашумленное изображение `'moonview08.tif'`, строим его спектр из которого видно, что шум кольцевой периодический. Потом строим режекторный кольцевой идеальный фильтр и с его помощью выполняем фильтрацию изображения в частотной области.

```
% удаление периодического шума кольцевым режекторным фильтром
close all;
g=imread('moonview08.tif');
imshow(g,[]);         %1 исходное зашумленное изображение
[M,N]=size(g);
G=fft2(g);
GG=fftshift(G);       % центрированное ПФ изображения
figure,imshow(abs(GG),[]);  %2 центрированный спектр
SF=log(1+abs(GG)/5000); % логарифмическое преобразование спектра
figure, imshow(SF,[]);  %3 улучшенное изображение спектра
% видим, что точки спектра шума расположены по кругу
% это значит, что шум периодический

% фильтрование кольцевым режекторным идеальным фильтром
D=@(u,v) u.^2+v.^2;
H=@(u,v,D0,W) 0+(1.*((D(u,v)<(D0-W/2).^2)|(D(u,v)>(D0+W/2).^2)));
```

```
[U,V]=meshgrid(-M/2:(M-1)/2,-N/2:(N-1)/2);
% 64 = угадываем радиус спектра периодического шума
HH=H(U,V,64,50);
figure,mesh(HH);  %3 поверхность кольцевого режекторного фильтра
HHB=fftshift(HH); %  децентрирование фильтра

% фильтрация в частотной области и ОПФ
FC=real(ifft2(G.*HHB'));    % G нецентрированное ПФ изображения
figure,imshow(FC,[]);        %4 восстановленное изображение
axis image;
```



На рисунке вверху слева показано исходное изображение, внизу слева – поверхность идеального режекторного фильтра, вверху справа – показано восстановленное изображение, внизу справа – спектр исходного изображения (белые точки шума увеличены для наглядности). На спектре хорошо видны частотные компоненты шума в виде пар симметричных ярких точек. Они приблизительно лежат на окружности с центром в начале координат частотной области, и, таким образом, использование кольцевого режекторного фильтра представляется вполне оправданным.

Замечание. Сценарий exPeridicNoise05.m создает зашумленное изображение и сохраняет его в файл `'moonview08.tif'` , а затем выполняет те же действия, что и сценарий приведенный выше.

На следующем рисунке приведены примеры графиков других кольцевых режекторных фильтров: слева показан идеальный фильтр, в середине – фильтр Баттерворта 1-го порядка, справа – гауссов фильтр.

Литература.

1. Р. Гонсалес, Р. Вудс Цифровая обработка изображений. М.: Техносфера, 2005.
2. Р. Гонсалес, Р. Вудс С. Эддинс Цифровая обработка изображений в среде Matlab. М.: Техносфера, 2006.
3. В.Т. Фисенко, Т.Ю. Фисенко, Компьютерная обработка и распознавание изображений: учеб. пособие. - СПб: СПбГУ ИТМО, 2008. –192 с.
4. Яне Б. Цифровая обработка изображений. Москва: Техносфера, 2007. - 584с.