МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования «Национальный
исследовательский университет «МИЭТ»

Кафедра: ВМ-1
Дисциплина: Математическое моделирование
01.03.04 «Прикладная математика» (выпускающая кафедра ВМ-1)

Курсовая работа на тему:
«Математическая модель фильтра Винера»

Выполнил:

студент группы ПМ-41

Алимагадов Курбан Алимагадович

Преподаватель:

доцент кафедры ВМ-1, к.ф.-м.н.

Лисовец Юрий Павлович

Дата:

МИЭТ 2019

Содержание

1. Вступление

В 1942 г. во время Второй мировой войны перед американским математиком Н. Винером встала задача отделения полезного сигнала от шума при решении задач автоматизации систем противовоздушной обороны, использующих радиолокационную технику. Винер решил задачу теоретически, разработав линейный фильтр, минимизирующий среднеквадратическую ошибку между выходным сигналом и полезной составляющей входного сигнала. Однако созданные на основе этой математической модели экспериментальные устройства не нашли применения в реальных системах ПВО, из-за малой производительности, недостаточной скорости работы и в силу ряда других причин.

Однако мощности современных ЭВМ позволяют применять данный фильтр при сравнительно небольших временных затратах, он подходит для фильтрации не только радиосигналов, как планировалось изначально, но также для устранения шума и помех с видео, изображений, звуковых файлов и т. д.

2. Постановка задачи

В данной работе рассмотрена реализация фильтра Винера применительно к решению задачи фильтрации изображений. В качестве примеров рассмотрены ч/б изображения, однако вполне возможна фильтрация также и цветных изображений (RGB), при отдельной фильтрации каждого канала.

Задача: получить фильтр минимизирующий среднеквадратическую ошибку между выходным сигналом и полезной составляющей входного сигнала. Программная реализация должна обеспечивать увеличение пикового отношения сигнал/шум – PSNR (“Peak Signal-to-Noise Ratio”), измеряется в дБ, и структурной схожести исходного и восстановленного изображений – SSIM (“Structural Similarity”), измеряется в процентах, после фильтрации.

3. Идеализация модели

В рамках решаемой задачи будем предполагать, что на вход фильтра поступают изображения, содержащий только нормальный (гауссов) шум и периодическую (гармоническую) помеху. Эти два типа помех на изображениях наиболее распространены, шум имеющий гауссово распределение появляется на изображениях в результате воздействия множества факторов (перепады напряжения, паразитные токи и др.), причиной появления периодического шума являются электрические или электромеханические помехи во время получения изображения. Оба шума являются аддитивными, то есть значения пикселей изображения шума прибавляются к значения пикселей тестового изображения.

Описывая модель, будем рассматривать только принцип работы фильтра, не учитывая факторы, принимаемые во внимание, при физической реализации электронного устройства.

4. Использование законов и свойств для математического описания модели:

Фильтр Винера является частотным фильтром, то есть сначала с помощью дискретного преобразования Фурье (ДПФ) находится спектр изображения, происходит его фильтрация, затем с помощью обратного ДПФ мы переходим обратно из частотной области в пространственную и получаем восстановленное изображение.

Фильтрация в частотной области определяется как произведение функции-фильтра на преобразование Фурье исходного изображения. Пусть задан фильтр H(u,v), его эквивалентное представление в пространственной области соответствует взятию обратного преобразования Фурье $\mathfrak{F}^{-1}\{H(u,v)\} = h(x,y)$. H(u,v) также называют частотной характеристикой фильтра (ЧХ), а h(x,y) – импульсной характеристикой фильтра (ИХ).

Рассмотрим модель искажения одномерного сигнала (формула ЧХ фильтра Винера, которую мы получим в итоге, для одномерного и двумерного сигналов имеет аналогичный вид):

$$X(n) = Y(n) + V(n), \tag{1}$$

где все дискретные сигналы являются стационарными случайными процессами, причём шумовая помеха V(n) не коррелирована с полезным сигналом Y(n) и M[Y(n)] = M[V(n)] = 0, т. е.

$$cov\big(Y(n), V(n)\big) = M[Y(n)V(n)] - M[Y(n)]M[V(n)] = 0 \tag{2}$$

Так как M[Y(n)] = M[V(n)] = 0, то, следовательно, $M[Y(n)V(n)] = 0$.

Спектр

$$\widetilde{F_f}(v) = \sum_{n=-\infty}^{\infty} f(n)e^{-2\pi inv}$$

любого дискретного сигнала f(n) обладает следующим свойством:

$$\int_0^1 \left|\widetilde{F_f}(v)\right|^2 dv = \sum_{n=-\infty}^{\infty} |f(n)|^2 \tag{3}$$

Так как $\forall\, m \in \mathbb{Z}$: $\int_0^1 e^{2\pi imu}\, du = \begin{cases} 1, m = 0 \\ 0, m \neq 0 \end{cases}$, то интегрированием по отрезку $v \in [0,1]$ выражения

$$\left|\widetilde{F_f}(v)\right|^2 = \widetilde{F_f}(v)\overline{\widetilde{F_f}(v)} = \sum_{n=-\infty}^{\infty} f(n)e^{-2\pi inv} \sum_{k=-\infty}^{\infty} \overline{f(k)}e^{2\pi ikv} =$$

$$= \sum_{n=-\infty}^{\infty} f(n) \sum_{k=-\infty}^{\infty} \overline{f(k)}e^{2\pi i(k-n)v}$$

получим соотношение (3).

Усечём сигнал X(n) и возьмём только часть его отсчётов с номерами $n = 0, \pm 1, \dots, \pm L$, положив остальные равными нулю. Для такого фрагмента сигнала будем использовать обозначение

$$X_L(n) = \begin{cases} X(n), -L \le n \le L, \\ 0, \quad |n| > L. \end{cases}$$

На основании соотношения (3) можем записать:

$$\int_0^1 \left|\tilde{F}_{X_L}(v)\right|^2 dv = \sum_{n=-L}^{L} X^2(n) \quad (4)$$

где спектр дискретного сигнала $X_L(n)$

$$\tilde{F}_{X_L}(v) = \sum_{n=-L}^{L} X(n) e^{-2\pi i n v} \quad (5)$$

Мощность P, т. е. энергия, приходящаяся в среднем на один отсчёт равна:

$$P = \lim_{L\to\infty} \int_0^1 \frac{M\left(\left|\tilde{F}_{X_L}(v)\right|^2\right)}{2L+1} dv = \lim_{L\to\infty} \frac{\sum_{n=-L}^{L} M(X^2(n))}{2L+1} = M(X^2(n)) = \sigma_X^2 \quad (6)$$

Нетрудно показать, что

$$\tilde{s}_X(v) = \lim_{n\to\infty} \frac{1}{2L+1} M\left(\left|\tilde{F}_{X_L}(v)\right|^2\right), \quad (7)$$

где

$$\tilde{s}_X(v) = \sum_{k=-\infty}^{\infty} K_X(k) e^{-2\pi i k v} \quad (8)$$

– спектр мощности дискретного сигнала X(n), а

$$K_X(k) = cov\big(X(n), X(n-k)\big) = M[X(n)X(n-k)]$$

– его ковариационная функция (при условии, что M[X(n)] = 0).

Из (6) и (7) следует, что дисперсия σ_X^2 дискретного стационарного процесса X(n) связана с его спектром мощности (8) следующим соотношением

$$\sigma_X^2 = \int_0^1 \tilde{s}_X(v) dv \quad (9)$$

Спектр мощности дискретного сигнала X(n), как несложно видеть из (7) и (8), является вещественной и неотрицательной периодической функцией: $\tilde{s}_X(v) = \tilde{s}_X(v+1) \geq 0$.

Будем обозначать искомую частотную характеристику фильтра

$$W(n) = \sum\nolimits_n w(n)\, e^{-2\pi i n v},$$

где w(n) – его импульсная характеристика. Для сигнала (1) фильтр Винера должен обеспечивать формирование наилучшей оценки

$$\hat{Y}(n) = \sum_k X(n-k)w(k)$$

в смысле минимума мощности сигнала-ошибки $E(n) = \hat{Y}(n) - Y(n)$

$$\sigma_X^2 = M\left\{\left(Y(n) - \hat{Y}(n)\right)^2\right\} \to min \qquad (10)$$

Для имеющих нулевое математическое ожидание взаимно некоррелированных сигналов Y(n) и V(n) спектр мощности $\tilde{s}_E(v)$ сигнала-ошибки $E(n) = \hat{Y}(n) - Y(n)$ можно представить в следующем виде:

$$\tilde{s}_E(v) = |W(v)|^2\left(\tilde{s}_Y(v) + \tilde{s}_V(v)\right) + (1 - W(v) - \overline{W(v)})\tilde{s}_Y(v), \qquad (11)$$

где $\tilde{s}_Y(v)$ и $\tilde{s}_V(v)$ – спектры мощности полезного сигнала $Y(n)$ и шумовой помехи $V(n)$ соответственно.

С учётом (9) условие оптимальной фильтрации (10) определяет следующие требования к фильтру Винера в частотной области:

$$\sigma_E^2 = \int_0^1 |W(v)|^2\left(\tilde{s}_Y(v) + \tilde{s}_V(v)\right) + \left(1 - W(v) - \overline{W(v)}\right)\tilde{s}_Y(v) \to \min_{W(v)}. \qquad (12)$$

Необходимо найти такую ЧХ $W(v)$, которая обеспечивает минимальное значение σ_E^2.

Очевидно, что для обеспечения минимума значения интеграла в выражении (12) нужно выбирать ЧХ $W(v)$, в каждой точке $v \in [0,1]$ такой, чтобы подынтегральная функция принимала минимальное значение:

$$\tilde{s}_E(v) = |W(v)|^2\left(\tilde{s}_Y(v) + \tilde{s}_V(v)\right) + \left(1 - W(v) - \overline{W(v)}\right)\tilde{s}_Y(v) \to \min_{W(v)}. \qquad (13)$$

Обозначив $A = |W(v)|$ и $\varphi = \arg(W(v))$, несложно найти минимум функции двух вещественных переменных, которая получается в результате преобразования соотношение (13):

$$g(A,\varphi) = A^2\left(\tilde{s}_Y(v) + \tilde{s}_V(v)\right) + (1 - 2Acos(\varphi))\tilde{s}_Y(v) = \tilde{s}_E(v) \to \min_{A,\varphi}. \qquad (14)$$

Из условий $\frac{\partial g}{\partial A} = 0$ и $\frac{\partial g}{\partial \varphi} = 0$ получаем:

$$\begin{cases} A = \dfrac{\tilde{s}_Y(v)\cos(\varphi)}{\tilde{s}_Y(v) + \tilde{s}_V(v)} \\ \sin(\varphi) = 0 \end{cases}$$

Значение $\varphi = \pi$ дало бы модуль $A = |W(v)| < 0$ (поскольку $\tilde{s}_Y(v) \geq 0$, $\tilde{s}_V(v) \geq 0$), поэтому полагаем $\varphi = 0$ и в результате получаем для ЧХ фильтра Винера выражение:

$$W(v) = \frac{\tilde{s}_Y(v)}{\tilde{s}_Y(v) + \tilde{s}_V(v)} \tag{15}$$

Рассматриваемая реализация фильтра Винера проходит окном по изображению (это помогает более точно определить дисперсию на однородных участках), вырезая фрагменты картинки с весовой маской, чтобы избежать возникновения эффекта Гиббса на смежных границах соседних окон.

Маска представляет собой матрицу N x N, центральные элементы которой равны 1, элементы на границе – 0, промежуточные элементы, находящиеся между центральными и граничными, представляют собой монотонно убывающую последовательность значений от 1 до 0 (область “спуска”). Окно перемещается по изображению с шагом step = ones_len + vect_down_len, где ones_len – длина единичной части, а vect_down_len – длина промежуточной части. Таким образом, окна, а значит и маски будут перекрываться. Сумма любых перекрывающихся элементов соседних масок должна быть равна 1, чтобы не уменьшать/увеличивать яркость пикселей по сравнению с исходным изображением.

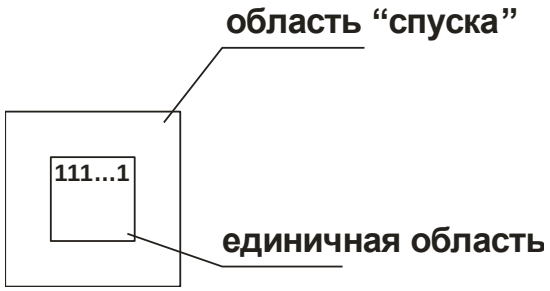

Рис. 1 Схема маски

5. Исследование математической модели:

Будем пользоваться маской размера 20 x 20 с единичной частью 8 x 8 пикселей.

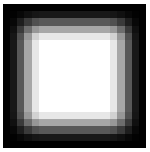

Рис. 2 Увеличенное изображение маски

Для применения фильтра к изображению или его фрагменту необходимо определить дисперсию шума (спектр мощности шума $\tilde{s}_V(v)$ из соотношения (15)) участка изображения, который будет фильтроваться.

Значение дисперсии шума, используемое для формирования фильтра Винера, находится как дисперсия значений пикселей фрагмента зашумлённого изображения. Для достижения более качественного результата фильтрации бывает полезно умножить этот параметр на коэффициент k_s (в работе на коэффициент умножается среднеквадратичное отклонение шума σ). Значение коэффициента k_s было выбрано равным 6.

Фильтрация проходит в 2 этапа: на первом удаляется гармоническая помеха, изображение фильтруется сразу целиком, фильтр помогает выделить на спектре картинки всплески соответствующие гармоникам, затем они вычитаются из спектра, после обратного ДПФ можно видеть, что периодическая помеха пропала; на втором этапе фильтруется нормальный шум, фильтр проходит окном по изображению, находит оценку среднеквадратичного отклонения значений пикселей, попавших в окно, и фильтрует этот фрагмент.

Для тестирования фильтра применялись стандартные изображения, используемые для проверки эффективности фильтров и алгоритмов обработки изображений: Lena, Barbara, Goldhill.

Рис. 4 Lena.png

Рис. 5 Barbara.png

Рис. 6 Goldhill.png

На изображения накладывается нормальный шум с математическим ожиданием m = 0 и среднеквадратическим отклонением σ = 20 и периодическая помеха, представляющая сумму нескольких гармоник.

Значения PSNR и SSIM до фильтрации:

Lena: PSNR_1 = 14.6, SSIM_1 = 0.15;

Barbara: PSNR_1 = 14.599, SSIM_1 = 0.244;

Goldhill: PSNR_1 = 14.618, SSIM_1 = 0.164.

Зашумлённые изображения (рис. 7-9):

Рис.7 Зашумлённое изображение Lena.png

Рис.8 Зашумлённое изображение Barbara.png

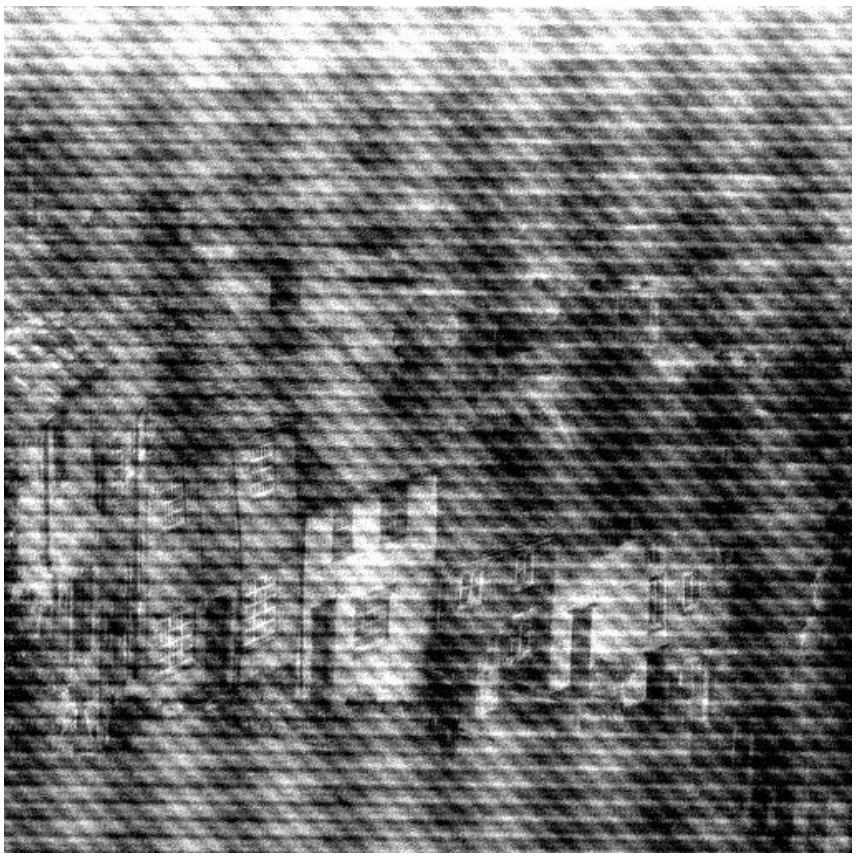

Рис.9 Зашумлённое изображение Goldhill.png

Результаты фильтрации:

Рис.10 Отфильтрованное изображение Lena.png

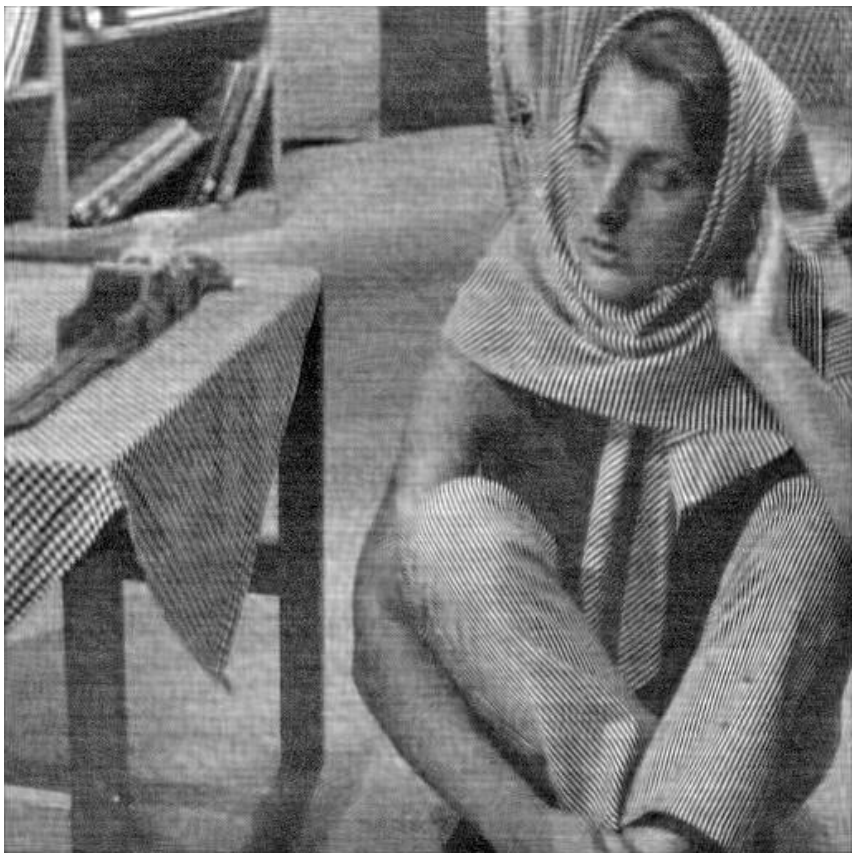

Рис.11 Отфильтрованное изображение Barbara.png

Рис.12 Отфильтрованное изображение Goldhill.png

Значения PSNR и SSIM до фильтрации:

Lena: PSNR_2 = 18.983, SSIM_2 = 0.411;

Barbara: PSNR_2 = 19.741, SSIM_2 = 0.54;

Goldhill: PSNR_2 = 18.118, SSIM_2 = 0.478.

Таблица № 1. Разности значений PSNR и SSIM после и до фильтрации.

Изображение	Lena	Barbara	Goldhill
Diff_PSNR, Дб	4,383	5,142	3,527
Diff_SSIM, %	0,261	0,296	0,314

Стоит отметить, что возможно увеличить эффективность фильтрации, в смысле увеличения значения PSNR и/или SSIM отфильтрованной картинки, если подбирать оптимальное значение параметра k_s индивидуально для каждого изображения.

Графики зависимости значений разностей PSNR и SSIM после и до фильтрации изображения Barbara.png от значений параметра k_s приведены на рис 13-14.

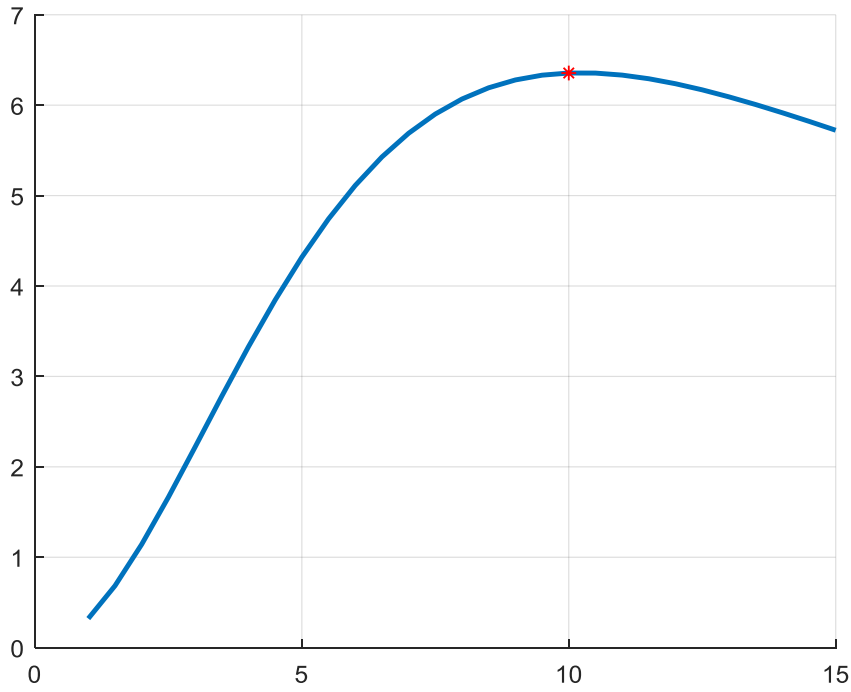

Рис.13 График Diff_PSNR(k_s)

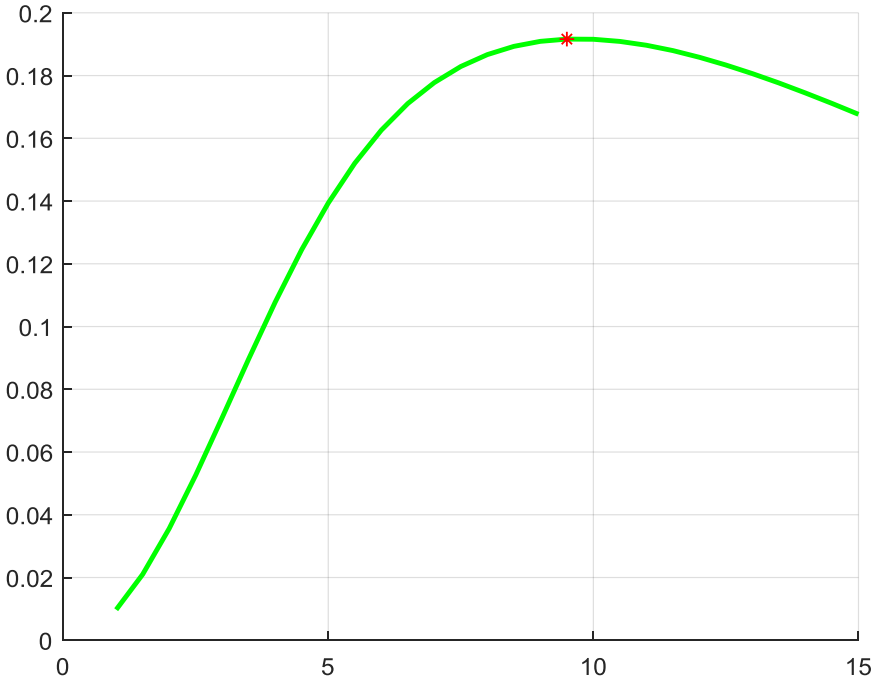

Рис.14 График Diff_SSIM(k_s)

На графиках отмечены точки максимума функций (k_s = 10, Diff_PSNR = 6.356) и (k_s = 9.5, Diff_PSNR = 0.1916). Таким образом, взяв значение k_s равным 9.75, мы можем получить наиболее эффективный результат фильтрации: мощность шума на изображении уменьшится больше чем на 6.3 Дб, а структурная схожесть с оригиналом увеличится почти на 20%.

6. Выбор переменных и констант

Основные переменные и константы, используемые при реализации математической модели в среде Matlab:

im – матрица исходного изображения;

sigma – среднеквадратическое отклонение шума;

noise – матрица шума;

im_n – матрица зашумлённого изображения;

PSNR_1 – значение PSNR до фильтрации;

PSNR_2 – значение PSNR после фильтрации;

SSIM_1 – значение SSIM до фильтрации;

SSIM_2 – значение SSIM после фильтрации;

mask_size – длина и ширина квадратной маски;

ones_size – длина единичной части маски;

vect_down_len – длина вектора “спуска”;

im_frame – зашумлённое изображение с рамкой (рамка необходима для правильного прохода окна с маской по изображению);

sp_window – спектр изображения/(фрагмента изображения вырезанного окном);

sigma_f – среднеквадратического отклонение шума;

k_s – коэффициент, на который умножается sigma_f;

Sg – спектр мощности зашумлённого изображения;

Sf – оценка спектра мощности исходного изображения;

Filt_winn – матрица ЧХ фильтра Винера;

im_res – матрица отфильтрованного изображения с рамкой;

Diff_PSNR – разность значений PSNR после и до фильтрации;

Diff_SSIM – разность значений SSIM после и до фильтрации;

7. Код на языке Matlab

Функции формирующие шум:

```
function R = imnoise2(type, M, N, a, b)
if nargin == 1
    a = 0; b = 1;
    M = 1; N = 1;
elseif nargin == 3
a = 0; b = 1;
end
switch lower(type)
case 'uniform'
R = a + (b - a)*rand(M, N);
case 'gaussian'
R = a + b*randn(M, N);
case 'salt & pepper'
if nargin <= 3
a = 0.05; b = 0.05;
end
if (a + b) > 1
    error('The sum Pa + Pb must not exceed 1.')
end
R(1:M, 1:N) = 0.5;
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 'rayleigh'
        R = a + (-b*log(1 - rand(M, N))).^0.5;
    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

function [r, R, S] = imnoise3(M, N, C, A, B)
[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
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);
    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
S = abs(R);
r = real(ifft2(ifftshift(R)));
end
```

Вспомогательные функции, формирующие маску и рамку для изображения:

```
function mask = SquareMask(size_mask,size_ones,func,vy,flag)
if (size_ones > size_mask)
    error('Длина единичной части должна быть меньше длины
маски!')
end
if (mod(size_ones, 2) ~= mod(size_mask, 2))
    error('Чётность длин маски и единичной части должна
совпадать!')
end
if (flag)
    vect_down = vy;
    i1 = (size_mask - size_ones)/2;
    i2 = (size_mask + size_ones)/2;
    if (i1 ~= length(vect_down))
        error('Неверно задана длина вектора спуска!')
    end
else
    vect_down_len = (size_mask - size_ones)/2;
    i1 = vect_down_len;
    i2 = size_mask - vect_down_len;
    step = 1 / (vect_down_len - 1);
    vx = 0:step:1;
    vect_down = func(vx);
end
lent_mask = zeros(1,size_mask);
lent_mask(i1 + 1:i2) = 1;
vect_down = sort(vect_down);
lent_mask(1:i1) = vect_down(:);
lent_mask(i2 + 1:end) = vect_down(end:-1:1);
mask = lent_mask' * lent_mask;
f_test = lent_mask(1: i1) + lent_mask(i2 + 1: end);
f_test = sum(f_test)/length(f_test);
if (f_test ~= 1)
    error('Сумма перекрывающихся частей не равна 1!')
end
end
```

```
function [im_frame, n1, n2] = ImFrame(im,mask,vect_down_len)
n1 = (size(im,1) + vect_down_len)/(size(mask,1) -
vect_down_len);
n2 = (size(im,2) + vect_down_len)/(size(mask,2) -
vect_down_len);
if ((n1 ~= floor(n1)) || (n2 ~= floor(n2)))
    n1
    n2
    error('Маска не подходит для обработки данного
изображения!')
end
i = size(im,1) + 2*vect_down_len;
j = size(im,2) + 2*vect_down_len;
im_frame = double(zeros(i,j)); % изображение с рамкой
i1 = 1 + vect_down_len;
i2 = size(im_frame,1) - vect_down_len;
j1 = 1 + vect_down_len;
j2 = size(im_frame,2) - vect_down_len;
im_frame(i1:i2,j1:j2) = im;
end
```

Функция, формирующая фильтр Винера и фильтрующая изображение:

```
function im_res =
Filtr_Winn3(im_frame,mask,vect_down_len,k_m,k_s)
i1 = 1 + vect_down_len;
i2 = size(im_frame,1) - vect_down_len;
j1 = 1 + vect_down_len;
j2 = size(im_frame,2) - vect_down_len;

im_n = im_frame(i1:i2,j1:j2);% зашумлённое изображение
im_res = zeros(size(im_frame));
step = size(mask,1) - vect_down_len;
for i = 0:step:size(im_n,1) % в двойном цикле проходим окном по
изображению и находим спектр окна
    for j = 0:step:size(im_n,2)

        i1 = i + 1;
        i2 = i + size(mask,1);
        j1 = j + 1;
        j2 = j + size(mask,2);

        im_window = double(im_frame(i1:i2,j1:j2)) .* mask;
        im_temp = zeros(k_m.*size(mask));
        i1 = (size(im_temp,1) - size(im_window,1))/2;
        i2 = (size(im_temp,1) + size(im_window,1))/2;
        j1 = (size(im_temp,2) - size(im_window,2))/2;
        j2 = (size(im_temp,2) + size(im_window,2))/2;
        im_temp(i1 + 1:i2,j1 + 1:j2) = im_window;

        sp_window = fftshift(fft2(im_temp));

        sigma_f = k_s*sqrt(var(im_window(:)));
```

```
        Sg = sp_window .* conj(sp_window); % спектр мощности
зашумлённого изображения
        Sf = Sg - sigma_f^2; % оценка спектра мощности исходного
изображения (могут получиться значения Sf < 0)

        for k = 1:size(Sf,1) % зануляем Sf < 0
            mas = find(Sf(k,:) < 0);
            for l = 1:length(mas)
                Sf(k,mas(l)) = 0;
            end
        end

        Filt_winn = Sf ./ Sg; % формируем фильтр Виннера для
окна
        Filt_winn(size(sp_window,1)/2 + 1,size(sp_window,2)/2 +
1) = 1;
        sp_window = Filt_winn .* sp_window; % применяем фильтр к
окну
        im_temp = ifft2(ifftshift(sp_window)); % переходим во
временную область и прибавляем полученный результат к матрице
восстановленного изображения
        im_window = im_temp(i1 + 1:i2,j1 + 1:j2);

        i1 = i + 1;
        i2 = i + size(mask,1);
        j1 = j + 1;
        j2 = j + size(mask,2);

        im_res(i1:i2,j1:j2) = im_res(i1:i2,j1:j2) + im_window;
    end
end

end
```

Скрипт, формирующий зашумлённые изображения и осуществляющий фильтрацию:

```
clc
clear
close all
% Первый этап
im = imread('Goldhill.png');
im = double(im);
sigma = 20;
noise = imnoise2('gaussian', 512, 512, 0, sigma); % генерируем
шум
im_n = im + noise; % формируем зашумлённое изображение

C = [20 4; -100 1; 32 -32; -50 1; -1 8;]; % координаты частот,
составляющих гарм. помехи на спектре относительно центра
изображения
A(1:size(C,1)) = [5000000 6000000 7000000 8000000 9000000 ]; %
амплтитуды составл. гарм. помехи
```

```
[r, R, S] = imnoise3(512, 512, C, A);
im_n = im_n + r;
MSE_1 = sum(sum((im - im_n).^2)) / (size(im,1) * size(im,2)); %
среднеквадратичное отклонение значений пикселей восстановленного
и исходного изображений
PSNR_1 = 20*log(255/sqrt(MSE_1))/log(10)
SNR_1 = snr(im,im_n - im)
SSIM_1 = ssim(im,im_n)
figure
imshow(uint8(im_n))
title('Изображение с наложенной гарм. помехой')

k_s = 17000; % подобранное значение обеспечивает формирование
фильтра, выделяющего на спектре частоты периодического шума
sp_window = fftshift(fft2(im_n));
sigma_f = k_s*sqrt(var(im_n(:)));
Sg = sp_window .* conj(sp_window); % спектр мощности
зашумлённого изображения
Sf = Sg - sigma_f^2; % оценка спектра мощности исходного
изображения (могут получиться значения Sf < 0)

for k = 1:size(Sf,1) % зануляем Sf < 0
    mas = find(Sf(k,:) < 0);
    for l = 1:length(mas)
        Sf(k,mas(l)) = 0;
    end
end

Filt_winn = Sf ./ Sg; % формируем фильтр Виннера для окна
Filt_winn = 1 - Filt_winn; % инвертируем фильтр, чтобы он не
пропускал частоты периодического шума
Filt_winn(size(sp_window,1)/2 + 1,size(sp_window,2)/2 + 1) = 1;

sp_window = Filt_winn .* sp_window; % применяем фильтр к окну

im_res = ifft2(ifftshift(sp_window)); % переходим во временную
область и прибавляем полученный результат к матрице
восстановленного изображения

% Второй этап

f = @(x)((cos(pi .* x) + 1) / 2).^1;
mask_size = 20;
ones_size = 8;
vect_down_len = (mask_size - ones_size) / 2;
mask = SquareMask(mask_size,ones_size,f,0,0);

im_frame = ImFrame(im_res,mask,vect_down_len);

PSF = fspecial('gaussian',7,7);

im_frame = edgetaper(im_frame,PSF);
```

```
im_2 = Filtr_Winn3(im_frame,mask,vect_down_len,4,6);

i1 = vect_down_len + 1;
i2 = size(im_frame,1) - vect_down_len;
j1 = vect_down_len + 1;
j2 = size(im_frame,2) - vect_down_len;
im_2 = im_2(i1:i2,j1:j2); % убираем рамку

figure
imshow(uint8(im_2))
title('Отфильтрованное изображение')
MSE_2 = sum(sum((im - im_2).^2)) / (size(im,1) * size(im,2)); %
среднеквадратичное отклонение значений пикселей восстановленного
и исходного изображений
PSNR_2 = 20*log(255/sqrt(MSE_2))/log(10)
SNR_2 = snr(im,im_2 - im)
SSIM_Win = ssim(im,im_2)
Diff_PSNR = PSNR_2 - PSNR_1
Diff_SNR = SNR_2 - SNR_1
Diff_SSIM = SSIM_Win - SSIM_1
```

8. Выводы

В данной работе была рассмотрена модель фильтра Винера для решения задачи фильтрации изображений. Полученный фильтр продемонстрировал уменьшение мощности шумовой составляющей сигнала после фильтрации и увеличение структурной схожести отфильтрованного изображения с оригиналом, по сравнению с неотфильтрованным.

Варьируя коэффициент k_s, на который умножается оценка среднеквадратического отклонения шума, являющаяся одним из параметров фильтра, можно найти его оптимальное значение, обеспечивающее наибольшее увеличение PSNR и/или SSIM после фильтрации.