Федеральное агентство по образованию
Государственное образовательное учреждение
высшего профессионального образования
САМАРСКИЙ ГОСУДАРСТВЕННЫЙ
АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
имени академика С.П. КОРОЛЕВА
Факультет информатики
Кафедра технической кибернетики

Выпускная квалификационная работа бакалавра

на тему

Разработка алгоритмов нелинейной фильтрации на основе идентификации линейных по параметрам моделей

Дипломник _____ Елкин Д.А.
(*подпись*)
Руководитель работы _____ Фурсов В.А.
(*подпись*)
Нормоконтролёр _____ Суханов С.В.
(*подпись*)

САМАРА 2014

Федеральное агентство по образованию
Государственное образовательное учреждение
высшего профессионального образования
САМАРСКИЙ ГОСУДАРСТВЕННЫЙ
АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
имени академика С.П. КОРОЛЕВА
Факультет информатики
Кафедра технической кибернетики

«УТВЕРЖДАЮ»
Заведующий кафедрой

_____
«___»_____ 20____ г.

ЗАДАНИЕ НА
ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА
студенту 6408 С202 группы ***Елкиу Денису Алексеевичу***

1 Тема работы ***Разработка алгоритмов нелинейной фильтрации на основе идентификации линейных по параметрам моделей***
утверждена приказом по университету от «14» марта 2014 г. № 96-ст
2 Исходные данные к работе:
постановка задачи.
3 Перечень вопросов, подлежащих разработке:
3.1 *анализ проблем улучшения качества изображений;*
3.2 *разработка технологии построения нелинейных фильтров на основе идентификации линейной по параметрам модели;*
3.3 *проведение экспериментальных исследований по восстановлению изображений с помощью разработанного алгоритма.*

Срок представления законченной работы «___» _____ 20___ г.

Руководитель работы _____ Фурсов В.А.
(*подпись*)
Задание принял к исполнению «___» _____ 20___ г.

_____ Елкин Д.А.
(*подпись*)

РЕФЕРАТ

Выпускная квалификационная работа бакалавра: 42 с., 24 рисунка, 10 источников, 2 приложения.

Презентация: 12 слайдов Microsoft PowerPoint.

НЕЛИНЕЙНАЯ ФИЛЬТРАЦИЯ ИЗОБРАЖЕНИЙ; ЗАДАЧА ИДЕНТИФИКАЦИИ ЛИНЕЙНОЙ ПО ПАРАМЕТРАМ МОДЕЛИ; ВОССТАНОВЛЕНИЕ ИЗОБРАЖЕНИЙ

Объектом исследования являются изображения, подвергнутые искажениям вида: размытие, расфокусировка, смаз.

Цель работы – разработать алгоритм идентификации параметров линейной модели для нелинейной фильтрации.

Разработан и программно реализован алгоритм восстановления искаженных изображений на основе нелинейной фильтрации, линейной по параметрам модели. Проведён ряд экспериментов, в ходе которых были получены сравнительные результаты использования линейных фильтров, перед нелинейными.

Содержание

ВВЕДЕНИЕ

Реставрацию можно рассматривать как процесс оценивания: некоторое изображение, полученное в результате наблюдения или измерения, подвергают преобразованию, чтобы найти оценку идеального изображения, которое наблюдалось бы на выходе гипотетической изображающей системы, не вносящей никаких искажений.

Для эффективного проектирования любой системы цифровой реставрации изображений необходимо знать количественные оценки искажений, вносимых физической искажающей системой, преобразователем в цифровую форму и дисплеем. Процедура реставрации в основном сводится к моделированию, а затем обращению искажающих преобразований [1].

В данной работе для моделирования искажающих преобразований был выбран алгоритм идентификации. Задача идентификации заключается в построении оптимальной или подходящей в некотором смысле модели системы (объекта) по результатам наблюдений входных и выходных переменных этого объекта. Построение модели обычно диктуется необходимостью изучения характеристик и закономерностей, присущих объекту. Для одного объекта могут быть построены различные формальные модели, отражающие различные аспекты его функционирования, изучаемые в рамках конкретных задач исследования [2].

Линейная фильтрация очень широко используется при устранении шумов на изображениях. Линейные КИХ-фильтры достаточно эффективны в вычислительном отношении и просты в реализации. Однако в приложении к цифровым изображениям они обладают рядом существенных недостатков: размывают границы и могут уничтожать мелко детальные особенности изображения. Эффект размывания границ может быть существенно снижен при использовании нелинейных фильтров [3].

Целью настоящей работы является разработка алгоритма нелинейной фильтрации на основе идентификации линейной по параметрам модели. А также написание программного для наглядного иллюстрирования качества восстановления изображений с помощью данного алгоритма.

В первом разделе описана формулировка задачи восстановления изображения и причины использования нелинейной фильтрации и алгоритма идентификации.

Во втором разделе описывается разработка технологии нелинейной фильтрации на основе алгоритма идентификации линейных по параметрам моделей. Рассматривается модель, описывается принцип ее работы и конкретная реализация модели.

В третьем разделе представлены практические результаты использования нелинейной фильтрации на основе алгоритма идентификации линейных по параметрам моделей.

1 Анализ проблем улучшения качества изображений, конкретизация задачи исследования

1.1 Формулировка задачи восстановления изображений

Пусть имеется полезный сигнал — последовательность $f(n)$. Однако непосредственному наблюдению (измерению) он недоступен. В нашем распоряжении имеется лишь сигнал $g(n)$ (результат прохождения сигнала через некоторую «искажающую» систему), дополнительно искаженный шумом $\nu(n)$ (см. рис. 1.1).

Рисунок 1.1. Модель наблюдения полезного сигнала

Требуется восстановить полезный сигнал по наблюдаемому. Для этого необходимо синтезировать такую восстанавливающую систему (фильтр), чтобы при подаче на ее вход наблюдаемого сигнала на выходе получалась бы оценка $\hat{f}(n)$ полезного сигнала (см. рис. 1.2) [3].

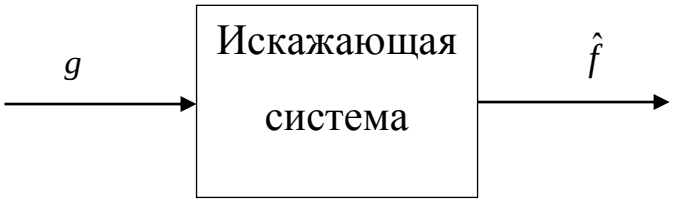


Рисунок 1.2 Схема восстанавливающей системы

1.2 Обзор существующих методов восстановления изображений

Уже существует множество методов восстановления (реставрации) изображений. К наиболее распространенным из них относятся: метод фильтрации Винера, инверсная фильтрация (метод преобразования Фурье), метод на основе статистического оценивания и др.

Исправленное изображение в случае инверсной фильтрации описывается в функцией:

$$\hat{F}_1(x,y) = F_1(x,y) + \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{N(\omega_x,\omega_y)}{H_D(\omega_x,\omega_y)}e^{i[\omega_x x,\omega_y y]}d\omega_x d\omega_y ,$$

где $F_1(x,y)$ — идеальное изображение; $H_D(\mathrm{x},\mathrm{y})$ — импульсный отклик линейной искажающей системы; $N(\mathrm{x},\mathrm{y})$ — аддитивный шум.

Частотная характеристика фильтра при винеровской фильтрации имеет вид:

$$H_H(\omega_x,\omega_y) = \frac{H_D^*(\omega_x,\omega_y)}{\left|H_D(\omega_x,\omega_y)\right|^2 + W_N(\omega_x,\omega_y)},$$

где $H_D(\omega_x,\omega_y)$ — частотная характеристика искажающей системы; $W_N(\omega_x,\omega_y)$ — энергетический спектр шума [1].

Отличительной чертой данных фильтров является необходимость знания о так называемом искажающем параметре системы (или искажающей функции). Так в случае фильтрации Винера и инверсной фильтрации, необходимо знать частотную характеристику искажающей системы.

1.3 Конкретизация задачи исследования

В данной дипломной работе в качестве построения восстанавливающего фильтра была взята за основу задача идентификации. Смысл данной задачи заключается в построение оптимальной или подходящей в некотором смысле модели системы (объекта) по результатам наблюдений входных и выходных переменных этого объекта Т.е. обратный искажающий оператор определяется путем идентификации параметров модели искажающей системы по специально сформированным для этой цели тестовым изображениям [2].

Линейная фильтрация очень широко используется при устранении шумов на изображениях. Линейные КИХ-фильтры достаточно эффективны в вычислительном отношении и просты в реализации. Однако в приложении к цифровым изображениям они обладают рядом существенных недостатков: размывают границы и могут уничтожать мелко детальные особенности изображения.

Эффект размывания границ может быть существенно снижен при использовании нелинейных фильтров [3]. Существуют и изучены следующие нелинейные фильтры для обработки изображений: Байесовская фильтрация [4]; Итерационные методы восстановления изображения [4]; Полиномиальный фильтр, характеризуемый дискретным функциональным рядом [5]; Медианная фильтрация [6]; Адаптивные фильтры [6]; Ранговая обработка изображений [6]; Пороговая фильтрация [7]; Фильтры экстремумов [7]. Но к сожалению ни один из низ не предназначен для восстановления размытых и расфокусированных изображений.

Итак описано преимущество использования алгоритма идентификации и использование нелинейной фильтрации. Далее будет рассмотрена реализация данного алгоритма для нелинейного фильтра.

2 Разработка технологии построения нелинейных фильтров на основе идентификации линейных по параметрам моделей

2.1 Линейная по параметрам модель нелинейного фильтра

Для решения ряда задач обработки (например восстановления) изображений необходимо знание оператора системы. Этот оператор рассчитывают на этапе проектирования с использованием физических и математических моделей видеотракта. Вследствие неизбежных упрощений, связанных как с недостаточной изученностью явлений, так и с вычислительными аспектами оператор системы нуждается в уточнении. Для этого проводят испытания, заключающиеся в «предъявлении» на вход системы известного изображения. Затем по входному изображению и результатам его регистрации на выходе системы строится оптимальная, в некотором смысле, модель. Эта задача известна как идентификация систем. Обычно она формулируется (в узком смысле [30]) как задача оценки параметров модели системы формирования изображений из априори заданного класса. Наиболее широко используются описанные линейные модели типа

$$g(n_1,n_2)=\sum_{m_1=-\infty}^{\infty}\sum_{m_2=-\infty}^{\infty}h(m_1,m_2)\,f(n_1-m_1,n_2-m_2) \qquad (2.1)$$

и/или

$$g(n_1,n_2)=\sum_{(m_1,m_2)\in Q_g}\sum a_{m_1,m_2}g(n_1-m_1,n_2-m_2)+\sum_{(m_1,m_2)\in Q_f}\sum b_{m_1,m_2}f(n_1-m_1,n_2-m_2) \qquad (2.2)$$

с постоянными по пространственным координатам параметрами (ЛПП-системы). Вычислительные преимущества, связанные с применением таких моделей, очевидны. Если в действительности искажения оказываются пространственно-зависимыми (неизопланатичными), то, как указывалось выше, линейные модели с постоянными параметрами строят на малых фрагментах изображений.

В данной работе за основу взята линейная по параметрам модель КИХ-фильтра (2.1), где вместо значений входного сигнала f выбираются равнозначные значения области f взятые нелинейно. Таким образом значения выходного сигнала рассчитываются нелинейным образом, а неизвестные коэффициенты модели входят в нее линейно. Например для линейного степенного ряда линейная по параметрам модель нелинейного фильтра будет иметь вид:

$$g(n_1,n_2)=\sum_{m_1=-\infty}^{\infty}\sum_{m_2=-\infty}^{\infty}\left(a^1_{m_1,m_2}f(n_1-m_1,n_2-m_2)+\right.$$
$$\left.+a^2_{m_1,m_2}f^2(n_1-m_1,n_2-m_2)+a^3_{m_1,m_2}f^3(n_1-m_1,n_2-m_2)+\ldots\right)$$

2.2 Задача идентификации линейного по параметрам фильтра

Для формулировки задачи идентификации указанных линейных моделей мы должны внести в уравнения (2.1), (2.2) некоторые важные дополнения. Во-первых, нельзя оценить бесконечное число значений импульсной характеристики, то есть число слагаемых в правой части (2.1) всегда ограничено. Во-вторых, задача идентификации решается по результатам измерений, которые всегда содержат погрешности. С учетом сказанного уравнения перепишем, соответственно, в виде

$$g(n_1,n_2)=\sum_{(m_1,m_2)\in Q_f}\sum h_{m_1,m_2}f(n_1-m_1,n_2-m_2)+\delta g(n_1,n_2), \tag{2.3}$$

$$\begin{aligned} g(n_1,n_2)&=\sum_{(m_1,m_2)\in Q_g}\sum a_{m_1,m_2}g(n_1-m_1,n_2-m_2)+\\ &+\sum_{(m_1,m_2)\in Q_f}\sum b_{m_1,m_2}f(n_1-m_1,n_2-m_2)+\xi(n_1,n_2). \end{aligned} \tag{2.4}$$

В (2.3), (2.4) используются те же обозначения, что и в (2.1), (2.2). Дополнительно введены лишь обозначения для ошибок $\delta g(n_1,n_2)$ (измерений, ограничений на порядок модели и др.), а фигурирующая в (2.4) ошибка $\xi(n_1,n_2)$ определяется как

$$\xi(n_1,n_2)=\delta g(n_1,n_2)-\sum_{(m_1,m_2)\in Q_g}\sum a_{m_1,m_2}\delta g(n_1-m_1,n_2-m_2). \tag{2.5}$$

Модель, описываемую уравнением (2.3), в соответствии с принятой терминологией далее будем называть КИХ-фильтром, а модель, соответствующую уравнению (2.4) — БИХ-фильтром. Поскольку задача идентификации должна решаться по совокупности отсчетов $f(n_1,n_2)$ входного (неискаженного) и $g(n_1,n_2)$ выходного (искаженного) изображений, для дальнейшего изложения удобно от уравнений (2.3), (2.4) перейти к их матричным представлениям.

Рассмотрим уравнение (2.4). Предположим, что общее число коэффициентов $\{a_{m_1,m_2}\}$, $\{b_{m_1,m_2}\}$ этого уравнения равно *M*, а фрагмент на выходном

изображении содержит N различных отсчетов $g(n_1, n_2)$. Отсчеты, фигурирующие в правой части (2.4) при одном из фиксированных положений (n_1, n_2) выходного отсчета, можно представить в виде элементов вектор-строки:

$$\mathbf{x}_i = \left[x_{i,1}, x_{i,2}, ..., x_{i,\mathrm{M}} \right] =$$
$$= \left[..., g(n_1 - m_1, n_2 - m_2), ..., f(n_1 - m_1, n_2 - m_2), ... \right], \quad (2.6)$$
$$(m_1, m_2) \in Q_g \cup Q_f, i = \overline{1, N},$$

а соответствующий вектор искомых параметров будет иметь вид

$$\mathbf{c} = \left[c_1, \mathrm{c}_2, ..., c_{\mathrm{M}} \right]^T = \left[..., a_{m_1,m_2}, ..., b_{m_1,m_2}, ... \right]^T, (m_1, m_2) \in Q_g \cup Q_f, i = \overline{1, N}. \quad (2.7)$$

Заметим, что размерность вектора искомых параметров зависит от порядка передаточной функции системы, то есть от размеров опорных областей как на входном, так и на выходном изображении. Для N различных положений опорных областей из векторов-строк (2.6) составим $N \times M$-матрицу $\mathbf{X}$, а из N отсчетов $g(n_1, n_2)$ на выходном изображении и N соответствующих им ошибок $\xi(n_1, n_2)$ составим $N \times 1$-векторы $\mathbf{y}$ и $\boldsymbol{\xi}$ соответственно. Если вектор параметров $\mathbf{c}$ остается неизменным при любом положении опорной области на фрагменте (для ЛПП-систем это всегда выполняется), с использованием введенных обозначений можно записать следующее матричное равенство:

$$\mathbf{y} = \mathbf{X}\mathbf{c} + \boldsymbol{\xi}. \quad (2.8)$$

Задача заключается в том, чтобы по одной реализации (фрагменту изображения) построить оценку $\hat{\mathbf{c}}$ вектора параметров $\mathbf{c}$ по доступным для непосредственного наблюдения $N \times M$-матрице $\mathbf{X}$ и $N \times 1$-вектору $\mathbf{y}(N > M)$, при неизвестном $N \times 1$-векторе ошибок $\boldsymbol{\xi}$.

Аналогичное матричное равенство можно построить для модели (2.3) КИХ-фильтра. Сопоставив приведенные выше обозначения с (2.3) нетрудно заметить, что вектор искомых параметров $\mathbf{c}$ в данном случае представляется в виде

$$\boldsymbol{c} = \left[c_1, c_2, ..., c_M\right]^T = \left[..., h_{m_1,m_2}, ...\right]^T, \left(m_1, m_2\right) \in Q_f$$

а каждая строка матрицы $\mathbf{X}$ состоит из отсчетов только входного изображения:

$$\boldsymbol{x}_i = \left[x_{i,1}, x_{i,2}, ..., x_{i,M}\right] = \left[..., f\left(n_1 - m_1, n_2 - m_2\right), ...\right], \left(m_1, m_2\right) \in Q_f, i = \overline{1, N}.$$

Постановка задачи идентификации модели КИХ-фильтра формально совпадает с задачей идентификации БИХ-фильтра. Важные отличия состоят в следующем. Компоненты $N \times 1$-вектора ошибок ξ в данном случае не зависят от полезных сигналов (отсчетов поля яркости выходного изображения), как это имеет место в (2.5) для БИХ-фильтра. Кроме того, размерность вектора искомых параметров зависит лишь от размеров опорной области на входном изображении. Это оказывается существенным для рассматриваемых методов. Подчеркнем, что с точки зрения задачи идентификации порядок обхода точек на фрагменте (изображении) не играет роли. Это приводит лишь к перестановке строк в уравнении (2.8). В то же время применение оцененных моделей КИХ и БИХ-фильтров существенно различается. [1]

2.3 Разработка алгоритмов идентификации двумерных фильтров

В данной дипломной работе рассматривается нелинейная фильтрация изображения с линейной по параметрам моделью на основе КИХ-фильтра. Для изучения была выбрана маска размером 5×5:

6	5	4	5	6
5	3	2	3	5
4	2	1	2	4
5	3	2	3	5
6	5	4	5	6

Таблица 1. Маска.

и функция вида:

$$f\left(x\right)=x+x^2+x^3 \tag{2.9}$$

Тогда для данной маски, уравнение выходного сигнала будет иметь следующий вид:

$$\begin{aligned} y = a_1x_1 + a_2x_2 + a_3x_2^2 + a_4x_2^3 + a_5x_3 + a_6x_3^2 + a_7x_3^3 + a_8x_4 + \\ +a_9x_4^2 + a_{10}x_4^3 + a_{11}x_5 + a_{12}x_5^2 + a_{13}x_5^3 + a_{14}x_6 + a_{15}x_6^2 + a_{16}x_6^3 \end{aligned}, \tag{2.10}$$

где $a_i, i=\overline{1,16}$ — идентифицируемые параметры, x_i — среднее значение яркости пикселей изображения с индексом соответствующим значению в маске (Таблица 1), т.е. равноудаленные от центра маски значения.

Таким образом первые три слагаемых выглядят следующим образом:

$$\begin{aligned} y = a_1x_{i,j} + a_2\frac{\left(x_{i-1,j-1}+x_{i-1,j+1}+x_{i+1,j+1}+x_{i+1,j-1}\right)}{4} + \\ +a_3\frac{\left(x_{i-1,j-1}+x_{i-1,j+1}+x_{i+1,j+1}+x_{i+1,j-1}\right)^2}{16} \end{aligned}. \tag{2.11}$$

Идентификация параметров модели происходит на основе исходного изображения и размытого. На исходном изображении случайным образом выбирается *N* точек, и выбирается набор соответствующих точек на размытом изображении. Далее строится система из *N* линейных уравнений, как

описано выше, где матрица $\mathbf{X}$ состоит из значений x_i из уравнения (2.10) для маски с центром в случайно выбранной точке, а вектор $\mathbf{y}$ состоит из значений яркости, случайно выбранных точек, исходного изображения. Производится проверка на наличие решения данной системы, согласно теореме Кронекера-Капелли: система линейных алгебраических уравнений совместна тогда и только тогда, когда ранг её основной матрицы равен рангу её расширенной матрицы, причём система имеет единственное решение, если ранг равен числу неизвестных и бесконечное множество решений, если ранг меньше числа неизвестных, т.е. для нашей системы должно выполняться следующее равенство:

$$rang\,\mathbf{X} = 16\,. \tag{2.12}$$

Затем по простой формуле находится оценка вектора решений $\hat{a}$:

$$\hat{a} = \left[\mathbf{X}^T\mathbf{X}\right]^{-1}\mathbf{X}^T\mathbf{y}\,. \tag{2.13}$$

Также осуществляется проверка критерия того, чтобы сумма значений вектора параметров было около 1; чтобы не сильно менялась яркость обрабатываемой области.

По найденным значениям параметров, и по формуле (2.9) можно восстанавливать изображения, применяя маску (Таблица 1), ко всем пикселям изображения.

2.4 Реализация технологии идентификации и восстановления

Для реализации алгоритма нелинейной фильтрации с линейной по параметрам моделью была реализована программа в математическом пакете Matlab. Алгоритм работы программы следующий: сначала считывается картинка (исходное изображение), затем из нее выбирается один канал цвета, чтобы дальше работать с черно-белой картинкой. Далее исходное изображение подвергается искажению – размытие. Во время выполнения дипломной работы было реализовано несколько алгоритмов размытия: размытие с помощью КИХ-фильтра, размытие с помощью БИХ-фильтра, фильтр низких частот Гаусса. Визуально было определено, что лучший визуальный эффект размытия дает фильтр низких частот Гаусса. Данный фильтр создается и применяется с помощью стандартных функции пакета Matlab: *fspecial('gaussian', hsize, sigma)* и *imfilter(image, filter, 'replicate')*. После этого запускается алгоритм идентификации параметров модели по искаженному и исходному изображению (описанный выше). После чего происходит попытка восстановления искаженного изображения, для оценки качества работы фильтра. Текст программы изложен в приложении Б.

Данный алгоритм и его реализация были отработаны и оттестированы на тестовых изображениях (миры, см. Приложение А). В ходе реализации программного средства были также например реализованы дополнительные функции, например такие как корректировка равномерности цветового баланса изображения.

Таким образом разработан и реализован алгоритм идентификации параметров линейной модели нелинейного фильтра для изображений. В следующей главе представлены практические результаты проделанной работы.

3 Экспериментальные исследования

3.1 Описание исходных данных и схемы эксперимента

Для тестирования и проверки программного средства были выбраны два основных тестовых изображения: мира (см. Приложение А) и фотография части Самарской области со спутника.

Реализована также программа, которая по найденным параметрам модели восстанавливает произвольно поданную картинку на вход. Принцип работы следующий: происходит считывание картинки (исходное изображение), затем из нее выбирается один канал цвета, чтобы дальше работать с черно-белой картинкой. Далее она подвергается искажению, аналогичному, что и при выполнении первой части: поиска параметров модели. Затем для известных параметров происходит попытка восстановления искаженной картинки. На протяжении работы программы выводятся сообщения об окончании каждого этапа обработки картинки, и каждая картинка выводится в отдельном окне, для возможности сравнения результата.

Также был реализован простейший линейный фильтр для возможности сравнения нелинейного фильтра с линейным. Линейный фильтр для той же маски, что и в случае нелинейного имеет вид:

$$y = a_1 x_1 + a_2 x_2 + ... + a_{25} x_{25} \quad (3.1)$$

где $a_i, i = \overline{1,25}$ - неизвестные параметры модели, $x_i, i = \overline{1,25}$ - значение яркости каждого пикселя маски. Для данного фильтра реализован аналогичный алгоритм нахождения вектора оценок параметров модели по исходному и искаженному изображениям. Текст программы изложен в приложении Б.

3.2 Результаты экспериментов

Рассмотри результат работы программы для различной силы размытия изображения (различное значение *sigma* при применении фильтра низких частот Гаусса).

Ниже проиллюстрированы результаты для слабого размытия (значение *sigma = 1.2*):

Вектор параметров нелинейного фильтра:

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
5.1902	-3.8189	0.0808	-0.0002	0.3009	-0.1165	0.0003	0.7376
a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}
-0.0414	0.0001	-2.7749	0.0988	-0.0003	1.357	-0.0216	7.5947e-05

Таблица 2. Значения вектора параметров нелинейного фильтра при слабом размытии.

Сумма значений равна *0.9919.*

Вектор параметров линейного фильтра:

a_1	a_2	a_3	a_4	a_5
6.3205	0.3811	2.1263	-0.2268	-0.4478
a_6	a_7	a_8	a_9	a_{10}
-2.1117	-3.019	-1.9179	-2.8602	-0.0817
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
-1.184	0.11526	0.5387	-0.2225	0.673
a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
0.105	0.0674	0.2323	0.1724	0.5715
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
-1.0903	-0.1789	0.5733	0.5594	-0.1548

Таблица 3. Значения вектора параметров линейного фильтра при слабом размытии.

Сумма значений равна *1.0026.*

Изображение 1. Исходное изображение.

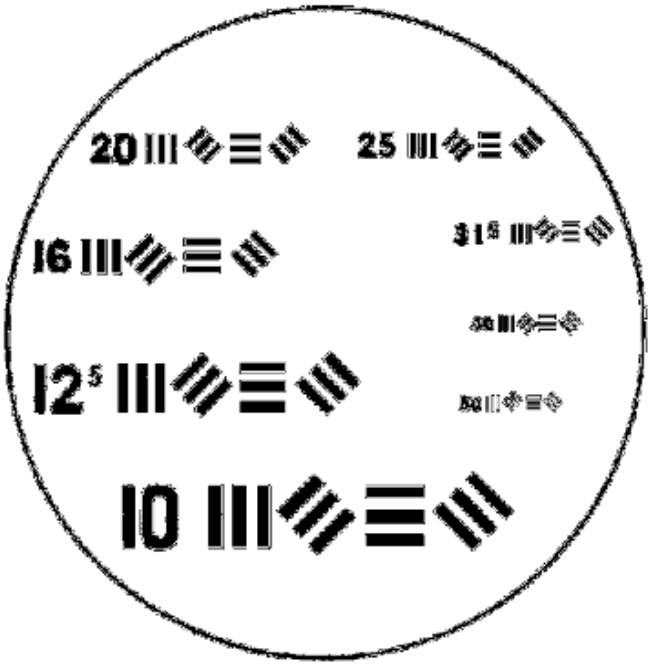
Изображение 2. Размытое изображение.

Изображение 3. Восстановленное с помощью нелинейного фильтра изображение.

Изображение 4. Карта разности между исходным и восстановленным изображениями.

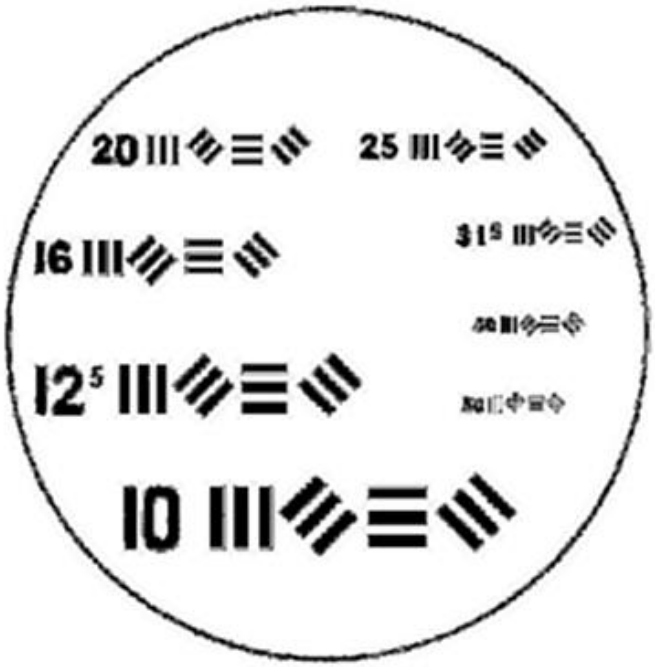
Изображение 5. Восстановленное с помощью линейного фильтра изображение.

Можно видеть, что нелинейный фильтр дает неплохой результат. Согласно карте разности между исходным и восстановленным изображениями мало заметная разница на краях объектов изображения. Но так же можно заметить, что очень мала разница между изображением восстановленным нелинейным фильтром и изображением восстановленным линейным фильтром. Далее проиллюстрированы результаты восстановления другого изображения, но для уже найденных параметров модели, т.е. задача идентификации параметров не решалась.

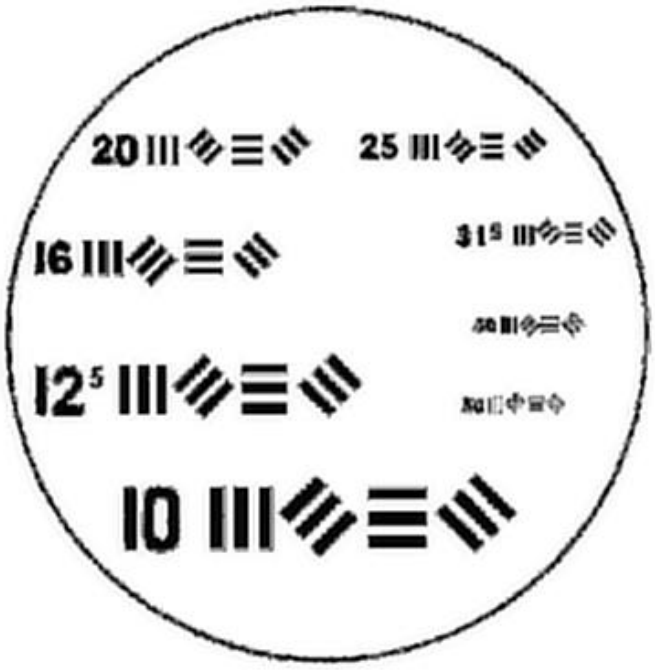


Изображение 6. Исходное изображение.

Изображение 7. Размытое изображение.



Изображение 8. Восстановленное с помощью нелинейного фильтра изображение.



Изображение 9. Восстановленное с помощью линейного фильтра изображение.

По представленным результатам хорошо видно, что для восстановления слабо размытых изображений фильтр работает очень хорошо, но незаметно разницы между линейным и нелинейным фильтрами.

Рассмотрим теперь результат для среднего размытия для таких же исходных изображений (значение параметра фильтра *sigma = 1.6*):

Вектор параметров нелинейного фильтра:

a_1	a_2	a_3	a_4
7.2442	-4.497	0.0739	-9.0658e-05
a_5	a_6	a_7	a_8
-2.4486	-0.05117	-1.8416e-05	0.6546
a_9	a_{10}	a_{11}	a_{12}
-0.0117	3.3038e-05	-0.395	-0.0546
a_{13}	a_{14}	a_{15}	a_{16}
0.0002	0.4408	0.0435	-0.0001

Таблица 4. Значения вектора параметров нелинейного фильтра при слабом размытии.

Сумма значений равна *0.9989.*

Вектор параметров линейного фильтра:

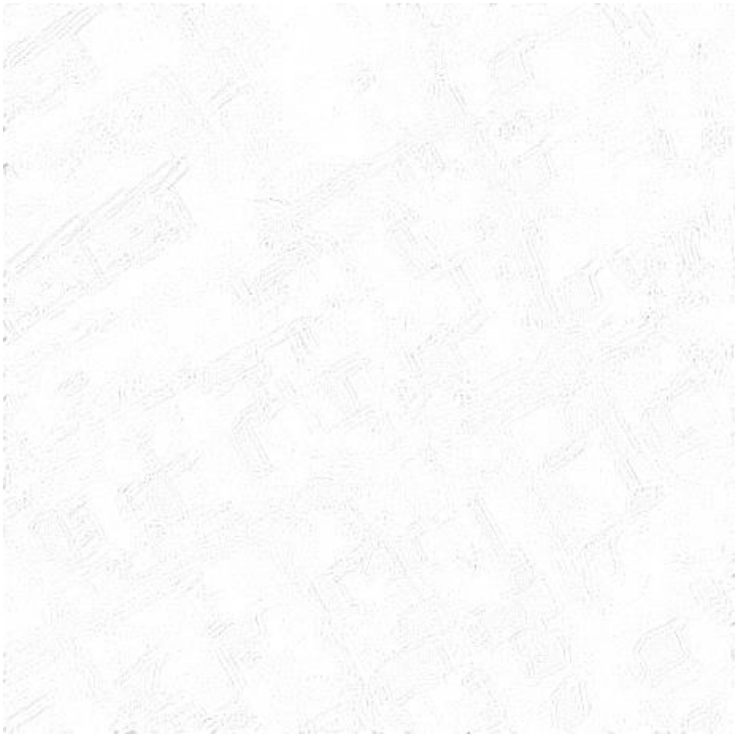
a_1	a_2	a_3	a_4	a_5
6.4343	0.2448	0.9279	0.8472	0.0715
a_6	a_7	a_8	a_9	a_{10}
-1.1118	-2.2484	-1.8928	-1.605	1.9866
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
-0.3214	-0.1973	0.8368	-1.0219	0.1008
a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
-1.0703	-1.0159	-2.5088	-1.2482	-0.4012
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
-0.0491	1.206	1.008	1.1151	0.9158

Таблица 5. Значения вектора параметров линейного фильтра при слабом размытии.

Сумма значений равна *1.0028.*

Изображение 10. Размытое изображение.

Изображение 11. Восстановленное с помощью нелинейного фильтра изображение.



Изображение 12. Карта разности между исходным и восстановленным изображениями.

Изображение 13. Восстановленное с помощью линейного фильтра изображение.

Визуально результат аналогичен случаю слабого размытия. Рассмотрим теперь результаты для работы фильтров с известными параметрами для другого изображения.

Изображение 14. Размытое изображение.

Изображение 15. Восстановленное с помощью нелинейного фильтра изображение.

Изображение 16. Восстановленное с помощью линейного фильтра изображение.

Для восстановления средне размытых изображений фильтр работает также хорошо, но разница между линейным и нелинейными фильтрами практически малозаметна. Лишь при пристальном рассмотрении увеличенного изображения можно увидеть, что нелинейный фильтр дает чуть более высокую четкость на краях объектов изображения.

Рассмотрим теперь результат для среднего размытия для таких же исходных изображений (значение параметра фильтра *sigma = 2*):

Вектор параметров нелинейного фильтра:

a_1	a_2	a_3	a_4
7.6485	-5.8515	0.1367	-0.0004
a_5	a_6	a_7	a_8
-0.1478	-0.0907	0.0003	5.1769
a_9	a_{10}	a_{11}	a_{12}
-0.0619	6.9113e-05	-7.2183	-0.0544
a_{13}	a_{14}	a_{15}	a_{16}
0.0002	1.4015	0.07	-0.0002

Таблица 6. Значения вектора параметров нелинейного фильтра при слабом размытии.

Сумма значений равна *1.0089.*

Вектор параметров линейного фильтра:

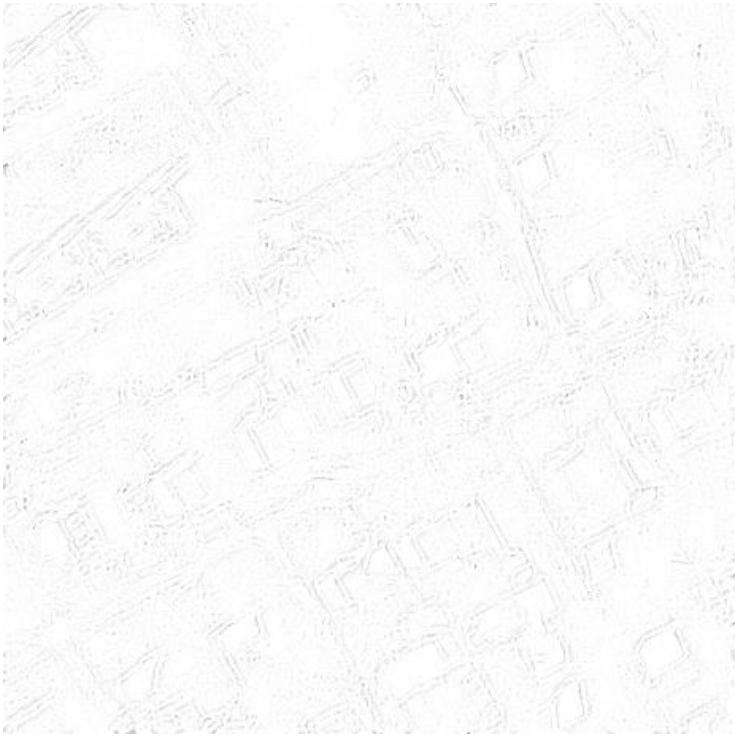
a_1	a_2	a_3	a_4	a_5
7.7786	2.9204	1.8288	3.081	2.5833
a_6	a_7	a_8	a_9	a_{10}
-3.0844	-3.208	-2.0024	-2.9293	-3.2013
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
0.1598	-0.01	-2.142	-2.5103	-0.19
a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
0.2858	-3.6973	-0.0708	-0.6806	-1.5225
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
-2.4002	1.3616	1.8153	3.0038	3.8354

Таблица 7. Значения вектора параметров линейного фильтра при слабом размытии.

Сумма значений равна *1.0046.*

Изображение 17. Размытое изображение.

Изображение 18. Восстановленное с помощью нелинейного фильтра изображение.



Изображение 19. Карта разности между исходным и восстановленным изображениями.

Изображение 20. Восстановленное с помощью линейного фильтра изображение.

Из результатов видно, что на карте разности отмечено намного больше ярких точек, чем ранее. А также более четко видно разницу между линейным и нелинейным фильтром: результат работы нелинейного фильтра выдал более четкую картинку, чем линейный фильтр. В результате чего можно судить о преимуществе нелинейного фильтра. Но прежде рассмотрим результаты для найденных параметров модели для другого изображения:

Изображение 21. Размытое изображение.

Изображение 22. Восстановленное с помощью нелинейного фильтра изображение.

Изображение 23. Восстановленное с помощью линейного фильтра изображение.

По представленным результатам можно судить о преимуществе использования нелинейного фильтра перед линейным. Границы объектов, при восстановлении с помощью не линейного фильтра более четкие, чем при восстановлении линейным фильтром.

Итак практически было показано достоинство использования нелинейной фильтрации с линейной по параметрам моделью.

ЗАКЛЮЧЕНИЕ

В данной работе была выполнена поставленная цель, а именно был разработан алгоритм нелинейной фильтрации на основе идентификации линейной по параметрам модели. Было также реализовано программное средство для реализации данного алгоритма и для визуального представления и оценивания результатов работы алгоритма.

Было теоретически описано преимущество использования алгоритма идентификации линейных по параметрам моделей, а также преимущество использования нелинейных фильтров. Был полностью описан процесс разработки алгоритма нелинейной фильтрации на основе идентификации линейной по параметрам модели, применимый для реставрации изображений. И в конце были представлены практические результаты использования данного алгоритма, перед линейными фильтрами.

Практически было установлено преимущество использования нелинейных фильтров перед линейными, что говорит о возможности использования их в реальных условиях, например для восстановления изображений со спутников.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Прэтт У. Цифровая обработка изображений. М.: Мир, 1982. Т. 1. 312 с.
2. Фурсов. Адаптивная идентификация по малому числу наблюдений.
3. Сойфер В. А. Методы компьютерной обработки изображений / Под ред. Сойфера В. А. 2-е изд., испр. М. Физматлит, 2003. 784 с.
4. Цифровая обработка изображений в информационных системах: Учебное пособие для студентов V курса РЭФ (специальности «Радиотехника» и «Средства связи с подвижными объектами») / И. С. Грузман, В. С. Киричук, В. П. Косых, Г.И. Перетягин, А. А. Спектор – Новосибирск: 2000. – 168 с.
5. Щербаков М. А. ИТЕРАЦИОННЫЙ МЕТОД ОПТИМАЛЬНОЙ НЕЛИНЕЙНОЙ ФИЛЬТРАЦИИ ИЗОБРАЖЕНИЙ // Известия ВУЗов. Поволжский регион. Технические науки. 2011. №4. С.43-56.
6. Теоретические основы цифровой обработки изображений/В. А. Сойфер, В. В. Сергеев, С. Б. Попов, В.В. Мясников – САМАРА 2000.- 256 с.
7. Давыдов А. В. Цифровая обработка сигналов: Тема 17. ОБРАБОТКА ИЗОБРАЖЕНИЙ / Электрон. текстовые данные – УГГУ – Режим доступа: http://prodav.exponenta.ru/dsp/index.html, свободный.
8. Даджион Д., Мерсеро Р. Цифровая обработка многомерных сигналов. М.: Мир, 1988. 488 с.
9. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде MATLAB. М.: Техносфера, 2006. 616 с.
10. Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.: Техносфера, 2005. 1072 с.

ПРИЛОЖЕНИЕ А

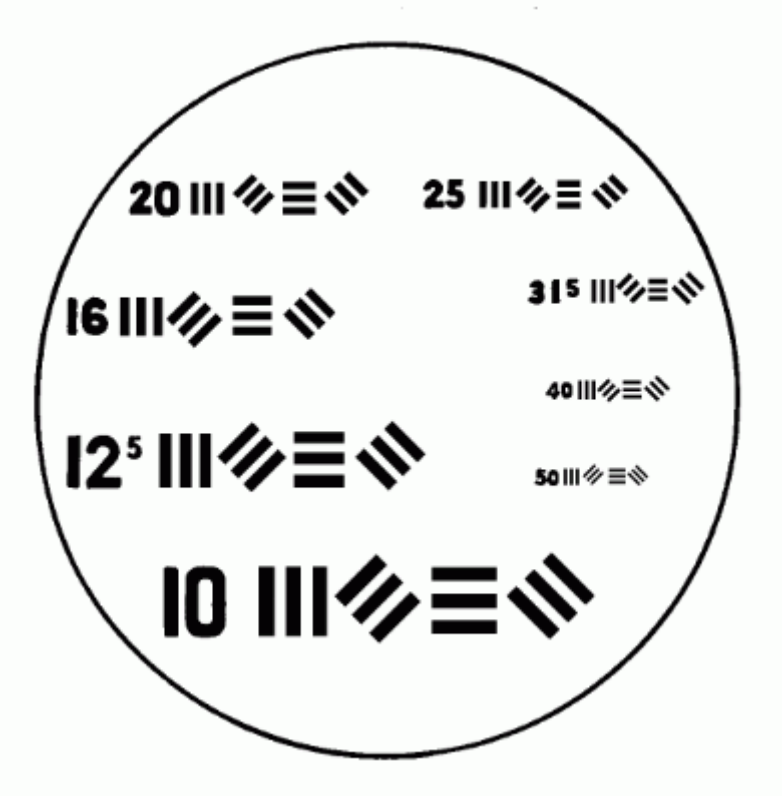


Рисунок 24. Тестовое исходное изображение миры.

ПРИЛОЖЕНИЕ Б

```
clear;
I = imread('cosmo1.jpg');
if length(I(1,1,:)) > 1
    temp_I = I(:,:,1);
    I=temp_I;
end
disp('now img monochrome');
H = fspecial('gaussian', length(I(:, 1)), 2);
blur_I = imfilter(I, H, 'replicate');
disp('img blurred');
figure;
imshow(blur_I)
n = 12;
a = zeros(1, n);
while sum(a) > 1.01 || sum(a) < 0.99
    a = final_solver(I, blur_I, 500);
end
disp('solver solved');
sum(a)
unblur_I = final_unblur(blur_I, a);
disp('img unblurred');
figure;
imshow(unblur_I)
dif = difference(I, unblur_I);
figure;
imshow(dif);

m = 25;
lin = zeros(1, m);
while sum(lin) > 1.1 || sum(lin) < 0.99
    lin = linear_solver(I, blur_I, m, 500);
end;
disp('linear solver solved');
sum(lin)
lin_unblur_I = unblur_lin(blur_I, lin);
disp('img unblurred by linear');
figure;
imshow(lin_unblur_I);


function [result] = final_solver(I, blur_I, samples_count)
    length_a = 16;
    new_I = double(I);
    new_blur_I = double(blur_I);
    X = zeros(samples_count, length_a);
    while rank(X) ~= length_a
        y = zeros(samples_count, 3);
        min_image_size = min(length(I(1,:)), length(I(:,1)));
        image_index_i = getRandom(3, min_image_size - 2, samples_count);
        image_index_j = getRandom(3, min_image_size - 2, samples_count);
        for i = 1 : samples_count
            y(i, 1) = new_I(image_index_i(i), image_index_j(i));
        end
        y(:, 2) = image_index_i;
        y(:, 3) = image_index_j;
        for i = 1 : samples_count
            src_i = y(i, 2);
            src_j = y(i, 3);
            b1 = new_blur_I(src_i, src_j);
```

```
            b2 = (new_blur_I(src_i + 1, src_j) + new_blur_I(src_i - 1, src_j)
+ new_blur_I(src_i, src_j + 1) + new_blur_I(src_i, src_j - 1)) / 4;
            b3 = (new_blur_I(src_i + 1, src_j - 1) + new_blur_I(src_i - 1,
src_j + 1) + new_blur_I(src_i + 1, src_j + 1) + new_blur_I(src_i - 1, src_j -
1)) / 4;
            b4 = (new_blur_I(src_i + 2, src_j) + new_blur_I(src_i - 2, src_j)
+ new_blur_I(src_i, src_j - 2) + new_blur_I(src_i, src_j + 2)) / 4;
            b5 = (new_blur_I(src_i + 1, src_j + 2) + new_blur_I(src_i + 1,
src_j - 2) + new_blur_I(src_i - 1, src_j + 2) + new_blur_I(src_i - 1, src_j -
2) + new_blur_I(src_i + 2, src_j - 1) + new_blur_I(src_i + 2, src_j + 1) +
new_blur_I(src_i - 2, src_j + 1) + new_blur_I(src_i - 2, src_j - 1)) / 8;
            b6 = (new_blur_I(src_i + 2, src_j - 2) + new_blur_I(src_i - 2,
src_j + 2) + new_blur_I(src_i + 2, src_j + 2) + new_blur_I(src_i - 2, src_j -
2)) / 4;
            X(i, :) = [b1 b2 b2^2 b2^3 b3 b3^2 b3^3 b4 b4^2 b4^3 b5 b5^2 b5^3
b6 b6^2 b6^3];
        end;
    end;
    y = y(:, 1);
    result = X \ y;
end


function [result] = final_unblur(blur_I, a)
    result = blur_I;
    temp = double(result);
    for i = 3 : length(blur_I(:,1)) - 2
        for j = 3 : length(blur_I(1,:)) - 2
            b1 = temp(i, j);
            b2 = (temp(i + 1, j) + temp(i - 1, j) + temp(i, j + 1) + temp(i,
j - 1)) / 4;
            b3 = (temp(i + 1, j - 1) + temp(i - 1, j + 1) + temp(i + 1, j +
1) + temp(i - 1, j - 1)) / 4;
            b4 = (temp(i + 2, j) + temp(i - 2, j) + temp(i, j - 2) + temp(i,
j + 2)) / 4;
            b5 = (temp(i + 1, j + 2) + temp(i + 1, j - 2) + temp(i - 1, j +
2) + temp(i - 1, j - 2) + temp(i + 2, j - 1) + temp(i + 2, j + 1) + temp(i -
2, j + 1) + temp(i - 2, j - 1)) / 8;
            b6 = (temp(i + 2, j - 2) + temp(i - 2, j + 2) + temp(i + 2, j +
2) + temp(i - 2, j - 2)) / 4;
            result(i, j) = round(b1 * a(1) + b2 * a(2) + b2^2 * a(3) + b2^3 *
a(4) + b3 * a(5) + b3^2 * a(6) + b3^3 * a(7) + b4 * a(8) + b4^2 * a(9) + b4^3
* a(10) + b5 * a(11) + b5^2 * a(12) + b5^3 * a(13) + b6 * a(14) + b6^2 *
a(15) + b6^3 * a(16));
        end
    end
end


I = imread('Mire.png');
if length(I(1,1,:)) > 1
    temp_I = I(:,:,1);
    I=temp_I;
end
disp('now img monochrome');
I = addLight(I);
disp('add light');
imshow(I);
disp('show src img');
% blur_I = blur(I, 0.1, 0.1, 0.2, 0.13, 0.13, 0.14, 0.3, 0.3);
H = fspecial('gaussian', length(I(:, 1)), 2);
```

```
blur_I = imfilter(I, H, 'replicate');
disp('img blurred');
figure;
imshow(blur_I)
unblur_I = final_unblur(blur_I, a);
disp('img unblurred');
figure;
imshow(unblur_I)

dif = difference(I, unblur_I);
figure;
imshow(dif);

lin_unblur_I = unblur_lin(blur_I, lin);
disp('img unblurred lin');
figure;
imshow(lin_unblur_I)


function [] = saver(prefix, I, blur_I, unblur_I, dif, lin_unblur_I)
    imwrite(I, strcat(prefix, '_src.png'));
    imwrite(blur_I, strcat(prefix, '_blur.png'));
    imwrite(unblur_I, strcat(prefix, '_unblur.png'));
    imwrite(dif, strcat(prefix, '_dif.png'));
    imwrite(lin_unblur_I, strcat(prefix, '_lun_unblur.png'));
end


function [result] = linear_solver(I, blur_I, length_a, samples_count)
    new_I = double(I);
    new_blur_I = double(blur_I);
    X = zeros(samples_count, length_a);
    while rank(X) ~= length_a
        y = zeros(samples_count, 3);
        min_image_size = min(length(I(1,:)), length(I(:,1)));
        image_index_i = getRandom(3, min_image_size - 2, samples_count);
        image_index_j = getRandom(3, min_image_size - 2, samples_count);
        for i = 1 : samples_count
            y(i, 1) = new_I(image_index_i(i), image_index_j(i));
        end
        y(:, 2) = image_index_i;
        y(:, 3) = image_index_j;
        for i = 1 : samples_count
            src_i = y(i, 2);
            src_j = y(i, 3);
            b1 = new_blur_I(src_i, src_j);
            b2 = [new_blur_I(src_i + 1, src_j) new_blur_I(src_i - 1, src_j)
new_blur_I(src_i, src_j + 1) new_blur_I(src_i, src_j - 1)];
            b3 = [new_blur_I(src_i + 1, src_j - 1) new_blur_I(src_i - 1,
src_j + 1) new_blur_I(src_i + 1, src_j + 1) new_blur_I(src_i - 1, src_j -
1)];
            b4 = [new_blur_I(src_i + 2, src_j) new_blur_I(src_i - 2, src_j)
new_blur_I(src_i, src_j - 2) new_blur_I(src_i, src_j + 2)];
            b5 = [new_blur_I(src_i + 1, src_j + 2) new_blur_I(src_i + 1,
src_j - 2) new_blur_I(src_i - 1, src_j + 2) new_blur_I(src_i - 1, src_j - 2)
new_blur_I(src_i + 2, src_j - 1) new_blur_I(src_i + 2, src_j + 1)
new_blur_I(src_i - 2, src_j + 1) new_blur_I(src_i - 2, src_j - 1)];
            b6 = [new_blur_I(src_i + 2, src_j - 2) new_blur_I(src_i - 2,
src_j + 2) new_blur_I(src_i + 2, src_j + 2) new_blur_I(src_i - 2, src_j -
2)];
            X(i, :) = [b1 b2 b3 b4 b5 b6];
```

```
        end;
    end;
    y = y(:, 1);
    result = X \ y;
end


function [result] = unblur_lin(blur_I, a)
    result = blur_I;
    temp = double(result);
    for i = 4 : length(blur_I(:,1)) - 3
        for j = 4 : length(blur_I(1,:)) - 3
            b1 = temp(i, j);
            b2 = [temp(i + 1, j) temp(i - 1, j) temp(i, j + 1) temp(i, j -
1)];
            b3 = [temp(i + 1, j - 1) temp(i - 1, j + 1) temp(i + 1, j + 1)
temp(i - 1, j - 1)];
            b4 = [temp(i + 2, j) temp(i - 2, j) temp(i, j - 2) temp(i, j +
2)];
            b5 = [temp(i + 1, j + 2) temp(i + 1, j - 2) temp(i - 1, j + 2)
temp(i - 1, j - 2) temp(i + 2, j - 1) temp(i + 2, j + 1) temp(i - 2, j + 1)
temp(i - 2, j - 1)];
            b6 = [temp(i + 2, j - 2) temp(i - 2, j + 2) temp(i + 2, j + 2)
temp(i - 2, j - 2)];
            result(i, j) = round([b1 b2 b3 b4 b5 b6] * a);
        end
    end
end


function [result] = difference(src_I, unblur_I)
    result = addLight(uint8(round(abs(255 - double(src_I) + dou-
ble(unblur_I)))));
end


function [result] = addLight(img)
    min_val = min(min(img));
    result = img - min_val;
    delta = max(max(img));
    for i = 1 : length(result(:, 1))
        for j = 1 : length(result(1, :))
            result(i, j) = round(double(img(i, j)) * 255 / delta);
        end
    end
end


function [result] = getRandom(min, max, cnt)
    result = randi([min, max], 1, cnt);
end
```