МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА
(национальный исследовательский университет)»

МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ПО ПРОВЕДЕНИЮ ЛАБОРАТОРНЫХ ЗАНЯТИЙ ПО ТЕХНОЛОГИЯМ ПРЕДПЕЧАТНОЙ ПОДГОТОВКИ ЦИФРОВЫХ ИЗОБРАЖЕНИЙ, РЕАЛИЗОВАННЫМ В СРЕДЕ MPI И CUDA С ИСПОЛЬЗОВАНИЕМ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ГЕТЕРОГЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ, ВКЛЮЧАЮЩИХ ГРАФИЧЕСКИЕ ПРОЦЕССОРЫ

Методические указания к лабораторным работам
«Исследование методов согласованной идентификации моделей цветовой коррекции с использованием суперкомпьютерных вычислительных технологий»,
«Устранение точечных бликов на цифровых изображениях»

САМАРА
2010

УДК 004.9

Составители:
Фурсов Владимир Алексеевич, Никоноров Артем Владимирович,
Кузьмишина Татьяна Михайловна, Бибиков Сергей Алексеевич,
Якимов Павел Юрьевич, Минаев Евгений Юрьевич,
Гайнуллина Гелия Мухаматкамиловна.

Методические указания включают описания двух лабораторных работ, посвященных изучению новых технологий цветовой коррекции и устранения артефактов в процессе предпечатной подготовки цифровых изображений, полученных путем регистрации цифровой копии с живописных полотен.

Лабораторные работы предназначены для студентов специальности “Технологии полиграфического производства” по курсу “Информатика” и для специалистов, проходящих курсы повышения квалификации.

Одобрены решением кафедры общей информатики ГОУ ВПО «Самарский государственный аэрокосмический университет имени академика С.П.Королёва (национальный исследовательский университет)», протокол № 4 от 25 ноября 2010 г.

УДК 004.9

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

«ИССЛЕДОВАНИЕ МЕТОДОВ СОГЛАСОВАННОЙ ИДЕНТИФИКАЦИИ МОДЕЛЕЙ ЦВЕТОВОЙ КОРРЕКЦИИ С ИСПОЛЬЗОВАНИЕМ СУПЕРКОМПЬЮТЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ ТЕХНОЛОГИЙ»

Составители:
С.А. Фурсов В.А., Никоноров А.В., Кузьмишина Т.М., Бибиков С.А.,
Якимов П.Ю., Минаев Е.Ю., Гайнулина Г.К.

Лабораторная работа предназначена для студентов специальности “Технологии полиграфического производства” по курсу “Информатика” и для специалистов, проходящих курсы повышения квалификации.
Одобрена решением кафедры общей информатики ГОУ ВПО «Самарский государственный аэрокосмический университет имени академика С.П.Королёва (национальный исследовательский университет)», протокол № 4 от 25 ноября 20010 г.

САМАРА 2010

Цель работы – изучение метода цветовой коррекции основанного на согласованной идентификации параметров модели коррекции. В лабораторной работе изучается метод цветовой коррекции цифровых изображений, основанный на согласованной идентификации.

1. Теоретические основы лабораторной работы.

1.1 Общая схема цветовой коррекции с использованием тестовых фрагментов.

Структурная схема автоматизированной системы цветовой коррекции на основе параметрической идентификации моделей цветовой коррекции по тестовым фрагментам может быть представлена в виде схемы (Рисунок 1).

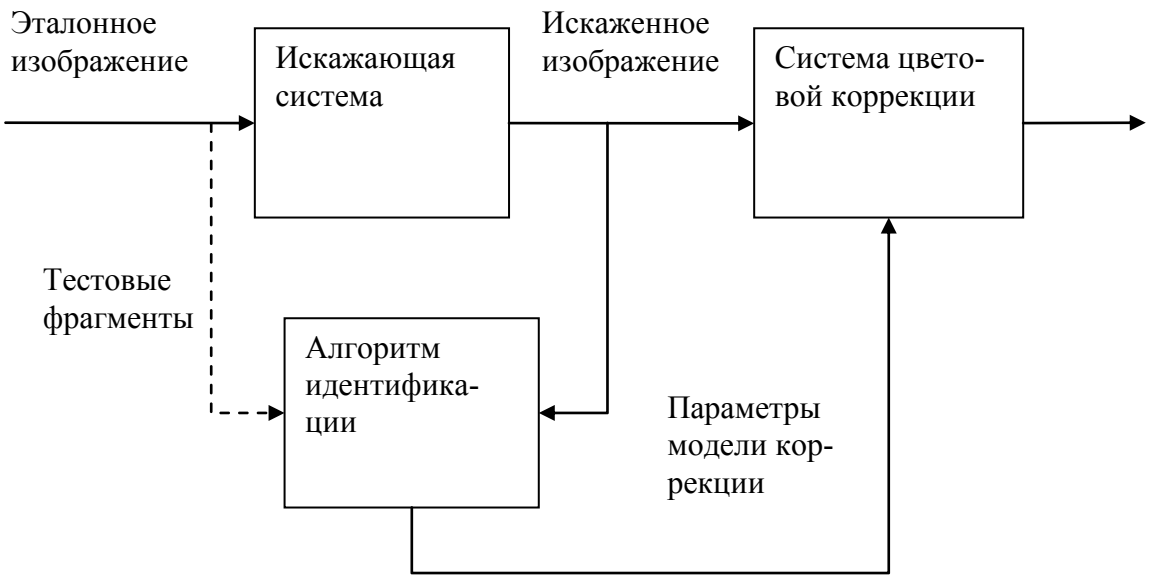

Рисунок 1 – Схема цветовой коррекции

Независимо от используемых для построения функций цветовой коррекции цветовых пространств, процедура формирования данных для идентификации модели цветового преобразования по тестовым фрагментам проводится следующим образом. Пользователь выбирает некоторую небольшую совокупность из N малых фрагментов, цвет которых может быть легко определен либо

с использованием субъективных представлений пользователя о цвете, либо с использованием в качестве эталонных цветов дополнительного изображения. Значение цвета на выделенном фрагменте изображения определяется усреднением цветов в области фрагмента:

$$I_n = \frac{1}{|E_n|}\sum I(x,y), \quad (x,y) \in E_n, n = \overline{1..N}. \quad (1)$$

Для каждого выбранного фрагмента задается желаемый (эталонный) цвет $I_n^* \quad n = \overline{1..N}$. В результате получается множество пар цветов

$$\left\{I_n, I_n^*\right\}, \quad n = \overline{1..N}. \qquad (2)$$

Рассмотрим решение общей задачи цветовой коррекции изображения, когда изменению подвергается все изображение, при этом модель цветовой коррекции можно записать в виде

$$\hat{I}(x,y) = I(x,y) - \hat{\psi}\left(I(x,y)\right).$$

Для правой части приведенного равенства строить модель цветовой коррекции изображений в цветовом пространстве RGB в виде соотношений, реализующих покомпонентные преобразования:

$$\begin{cases} I_r^* = \sum_{i=0}^{M} c_{ri}\left(I_r\right)^i, \\ I_g^* = \sum_{i=0}^{M} c_{gi}\left(I_g\right)^i, \\ I_b^* = \sum_{i=0}^{M} c_{bi}\left(I_b\right)^i, \end{cases} \quad (3)$$

где I_r, I_g, I_b – координаты цвета фрагмента на исходном изображении, I_r^*, I_g^*, I_b^* - «желаемые» координаты цвета на том же фрагменте, а c_{ri}, c_{gi}, c_{bi} - неизвестные коэффициенты, которые должны быть определены в результате решения задачи идентификации. Заметим, что составляющая $I(x,y)$, фигурирующая в модели цветовой коррекции, в соотношениях (3.3) учитывается коэффициентами c_{r0}, c_{g0}, c_{b0}.

Поскольку все три соотношения в (3) независимы и имеют одинаковый вид, ограничимся рассмотрением коррекции одного компонента цветового пространства. При этом модель ψ' цветовой коррекции для произвольной цветовой координаты можно представить в виде

$$I^* = \psi'(I) = \sum_{i=1}^{M} c_i (I)^i . \quad (4)$$

В силу модели (4) можно записать совокупность N соотношений вида:

$$I_n^* = \sum_{i=0}^{M} c_i (I_n)^i \quad \left(n = \overline{1..N}\right), \quad (5)$$

где I_n^* – значение цветовой координаты (R, G или B) на n-м фрагменте, «залитом» желаемым цветом; I_n – значение той же цветовой координаты на соответствующем фрагменте исходного изображения, а ξ_n – ошибки, связанные, в первую очередь, с неточным заданием желаемых цветов, субъективностью восприятия цветов, а также ошибками аппроксимации, а также с ошибками, связанными с неточностью задания параметрического класса модели и т.д.

В предположении неизменности параметров c_i модели (5) во всем цветовом пространстве совокупность соотношений (5) можно записать в виде переопределенной линейной системы в матричном виде:

$$\mathbf{Y} = \mathbf{X}^{\mathbf{T}}\mathbf{c} + \boldsymbol{\xi}, \quad (6)$$

где $\mathbf{Y} = \begin{bmatrix} I_1^* \\ I_2^* \\ \vdots \\ I_N^* \end{bmatrix}, \mathbf{X}^{\mathbf{T}} = \begin{bmatrix} 1 & I_1 & \cdots & (I_1)^M \\ 1 & I_2 & \cdots & (I_2)^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & I_N & \cdots & (I_N)^M \end{bmatrix}, \mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_N \end{bmatrix}, \boldsymbol{\xi} = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_N \end{bmatrix}.$

Далее решается задача идентификации модели цветовой коррекции (4) для каждой цветовой координаты в отдельности. Полученные покомпонентные модели затем используется для преобразования всех цветов исходного изображения.

При определении тестовых фрагментов на произвольном изображении следует руководствоваться следующими соображениями. Минимальный размер

фрагментов определяется возможным импульсным цветовым шумом, обусловленным особенностями регистратора. Максимальный размер определяется размерами области постоянного цвета. В настоящей работе, исходя из указанных соображений, использовались фрагменты размером 5×5 пикселей.

В результате реализации описанной выше процедуры идентификации могут быть получены кривые для преобразования всех трех координат цветового пространства. Однако вычисление с их использованием новых RGB-координат может привести к некорректному преобразованию цветов. Для обеспечения корректности преобразований необходимо выполнение требований, которые в случае цветовой коррекции в пространстве RGB формулируются в следующем виде:

Требование 1. Каждая кривая преобразования нормированного цветового RGB-пространства должна принадлежать квадрату со стороной равной единице:

$$\psi' : I \to I^* \quad I, I^* \in [0,1], \qquad (7)$$

притом $\psi'(0) = 0$, (8)

$\psi'(1) = 1$. (9)

Требование 2 Кривые преобразования цветов должны быть неубывающими:

$$\forall I_1, I_2 : I_1 > I_2 \to \psi'(I_1) \geq \psi'(I_2). \qquad (10)$$

Первое требование обеспечивает реализуемость цветов и корректные преобразования черного и белого цветов. Второе - позволяет избежать «инверсии» цвета, когда более яркая цветовая точка преобразуется в менее яркую.

Выполнение первого требования может быть обеспечено путем решения задачи идентификации модели (4) с «закрепленными» концами, т.е. при условии выполнения ограничений-равенств (8), (9). Если в результате идентификации указанной модели с ограничениями в точках «белого» и «черного» получается кривая, не удовлетворяющая второму условию, это условие проверяется в области меньшей, чем все цветовое пространство.

Для этого строится вписанный прямоугольник, заданных размеров. Если внутри этого прямоугольника требование *2* выполняется, точки пересечения кривой преобразования со сторонами вписанного прямоугольника, ближайшие к точкам с координатами (0,0) и (1,1), соединяются с этими точками прямыми линиями или квадратичными сплайнами (Рисунок 2). Пример получаемых кривых приведен на рисунке 3.

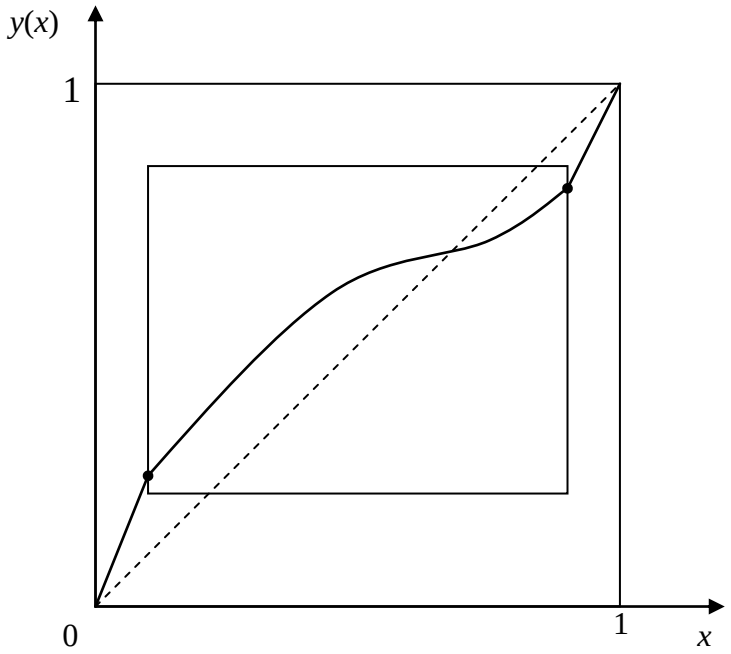

Рисунок 2 – Схематичное изображение построения кривой коррекции.

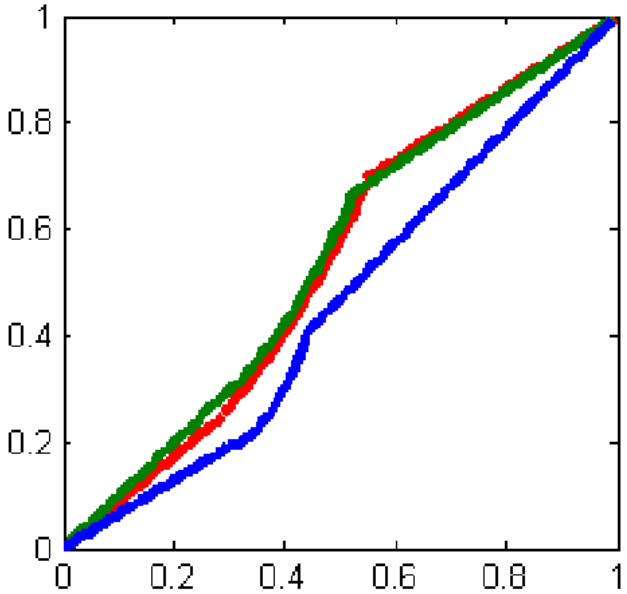

Рисунок 3 – Пример кривых коррекции.

Заметим, что если стороны вписанного прямоугольника слишком далеко отстоят от границ нормированного цветового пространства, это может приводить к значительной «потере» цветов и их искажению. Поэтому если применением близкого к квадрату с единичной стороной прямоугольника не удается обеспечить выполнение требования *3.2*, следует перейти к решению задачи оценивания модели (4) со «свободными» концами, т.е. без учета ограничений (8) и (9). Концы кривой преобразования как и ранее «закрепляются» путем соединения точек (0,0) и (1,1) с точками пересечения со сторонами вписанного прямоугольника.

Если решение задачи идентификации модели (4) со «свободными» концами не удовлетворяет требованию *3.2*, то решение ищется в параметрическом семействе моделей меньшего порядка. Порядок понижается до тех пор, пока не удастся выполнить требование *2* внутри заданного вписанного прямоугольника, достаточно близкого к единичному квадрату.

1.2 Параметрическая идентификация.

Один из основных этапов информационной технологии цветовой коррекции на основе идентификации – определение параметрического семейства моделей, описывающих вид преобразования цветовых координат или значений яркости. Параметрические классы моделей в основном определяются типом искажений и артефактов, вносимых осветительной и регистрирующей аппаратурой. В общем случае, когда параметрическое семейство моделей цветовой коррекции образовано нелинейными функциями, связь исходных и преобразованных цветовых координат с учетом (1.12) можно представить в виде:

$$z(x,y)=\psi\left(I(c,p,x,y)\right), \qquad (11)$$

где $z(x_i,y_i)$ – преобразованные значения цветовых координат, $\psi\left(I(c,p,x,y)\right)$ – функция преобразования исходных значений цветовых координат.

Наиболее простыми и удобными для решения задачи идентификации являются линейные или линейные по параметрам модели. Используя обозначения, принятые в (11), линейную модель цветовой коррекции для конкретных, наблюдаемых (заданных в виде желаемых) в точке (x_i, y_i) значений цветовых координат, можно представить в виде

$$z(x_i, y_i) = \sum_{j=1}^{M} c_j \cdot s_j(x_i, y_i) + \xi(i), \qquad (12)$$

где $s_j(x_i, y_i)$, $j = \overline{1, M}$ – значения заданной функции цветовых координат (или значения яркости) в точке (x_i, y_i).

Предположим теперь, что в рамках модели (12) для N различных фрагментов изображения $(N > M)$ наряду с известными значениями заданных функций цветовых координат $s_j(x_i, y_i)$, $j = \overline{1, M}$ каким-либо образом удалось задать также желаемые цветовые координаты или значения яркости $z(x_i, y_i), i = \overline{1, M}$. Из полученных таким образом N векторов-строк

$$\mathbf{s}_i = \left[s_1(x_i, y_i), s_2(x_i, y_i), ..., s_M(x_i, y_i) \right], i = \overline{1, N}$$

составим $N \times M$-матрицу $\mathbf{S}$, а из N желаемых (эталонных) значений яркости $z(x_i, y_i)$ и N соответствующих им ошибок $\xi(i)$ составим $N \times 1$-векторы $\mathbf{Z}$ и $\mathbf{\Xi}$ соответственно. Если предположить, что вектор параметров модели $\mathbf{c}$ остается неизменным для всех тестовых фрагментов изображения (т.е. функция цветового преобразования неизменна в соответствующей области цветового пространства), с использованием введенных обозначений можно записать следующее матричное равенство.

$$\mathbf{Z} = \mathbf{S}\mathbf{c} + \mathbf{\Xi}. \quad (13)$$

Задача идентификации заключается в том, чтобы по доступным для непосредственного наблюдения $N \times M$-матрице $\mathbf{S}$ и $N \times 1$-вектору $\mathbf{Z}$ $(N > M)$, сформированным с использованием совокупности заданных тестовых фрагментов изображения, построить оценку $\hat{\mathbf{c}}$ вектора параметров $\mathbf{c}$ при неизвестном $N \times 1$-векторе ошибок $\mathbf{\Xi}$.

Обычно, когда ставится задача построить оценку $\hat{\mathbf{c}}$ вектора параметров $\mathbf{c}$ по $N \times M$-матрице $\mathbf{S}$ и $N \times 1$-вектору $\mathbf{Z}$ $\left(N > M\right)$, связанным равенством (13), используют следующие предположения.

1. Векторы $\boldsymbol{\Xi} = \left[\xi_1, \xi_2, ..., \xi_N\right]^T$ и $\mathbf{Z} = \left[\mathrm{z}_1, \mathrm{z}_2, ..., \mathrm{z}_N\right]^T$ – случайные.
2. Матрица $\mathbf{S}$ детерминирована, т.е. ее элементы не являются случайными величинами.
3. $\mathrm{rank}\left(\mathbf{S}\right) = M$.
4. Математическое ожидание вектора $\boldsymbol{\Xi}$ равно нулю, т.е. $\mathrm{M}\left\{\xi_i\right\} = 0$, $i = \overline{1, N}$ или $\mathrm{M}\left\{\boldsymbol{\Xi}\right\} = 0$.
5. Для любых $\forall i, j : i \neq j$ $\mathrm{M}\left\{\xi_i \cdot \xi_j\right\} = 0$, $\mathrm{M}\left\{\xi_i^2\right\} = \sigma^2$ для всех $i = \overline{1, N}$. Другими словами, $\mathrm{cov}\left\{\boldsymbol{\Xi}\right\} = \sigma^2 \mathbf{E}_N$, где σ^2- дисперсия отклонений, $\mathrm{cov}\left\{\cdot\right\}$ – $N \times N$-матрица ковариаций отклонений, а $\mathbf{E}_N$ – единичная $N \times N$-матрица.

Если эти предположения выполняются, то оценка метода наименьших квадратов (МНК)

$$\hat{\mathbf{c}} = \left[\mathbf{S}^T \mathbf{S}\right]^{-1} \mathbf{S}^T \mathbf{Z} \quad (14)$$

является несмещенной и эффективной.

К сожалению, указанные предположения при малом числе наблюдений не отвечают реальному содержанию задачи и оказываются, в лучшем случае, бесполезными. Традиционно теория идентификации развивается в рамках теории статистического оценивания. В данном случае вследствие малого числа доступных тестовых фрагментов вероятностные модели сигналов оказываются ненадежными, т.к. не выполняется основное условие статистической устойчивости – существование большого числа наблюдений. При этом использование методов статистического оценивания оказывается малообоснованным. Свойство устойчивости статистических характеристик шумов при малом числе тестовых фрагментов изображений не проявляется даже в случае, когда существует

устойчивое распределение ошибок на множестве изображений данного класса. Поэтому в данном случае представляется целесообразным отказаться от всех наиболее важных с теоретической точки зрения предположений классической регрессии: $\mathrm{rank}(\mathbf{S}) = M$, $\mathrm{M}\{\boldsymbol{\Xi}\} = 0$ и $\mathrm{cov}\{\boldsymbol{\Xi}\} = \sigma^2 \mathbf{E}_{\mathrm{N}}$. Отказ от этих предположений является вынужденным, но оправданным.

В связи с последним замечанием нуждается в уточнении само понятие «малое число тестовых фрагментов». Как указывалось выше признаком, отличающим задачи оценивания по малому числу наблюдений от задач классической регрессии, следует считать нарушение условий статистической устойчивости. Однако свойство статистической устойчивости может не проявляться и при достаточно большом числе наблюдений. Более того, попытки указать конкретное число наблюдений, которое может считаться малым, неконструктивны. Выборка 500 наблюдений является достаточно представительной при оценивании одного параметра (сдвига), но это очень мало, если решается задача оценивания параметров разделяющей гиперплоскости в пространстве 400 признаков. Далее, для определенности, будем полагать число тестовых фрагментов малым, если их количество превышает число оцениваемых параметров модели цветовой коррекции не более чем в 2 раза.

Другая важная особенность задачи идентификации моделей цветовой коррекции – возможность появления грубых ошибок в задании желаемых цветов на тестовых фрагментах. В значительной степени это связано с субъективностью восприятия цвета. В частности, некоторые области цветового пространства могут восприниматься иначе, чем другие области того же пространства. Поэтому при построении алгоритмов идентификации должны быть предусмотрены меры по робастизации оценок параметров моделей цветовой коррекции.

Наиболее подходящим в рассматриваемой ситуации методом, по-видимому, является метод согласованной идентификации. Существо метода сводится к поиску подсистемы наиболее свободной от шума путем сравнения взаимной близости оценок, получаемых на множестве всех возможных совместных подсистем (с квадратными матрицами), сформированных на этих под-

системах. В рамках этого подхода удается выделить подмножество наиболее согласованных данных. Однако реализация этого метода «в чистом виде» связана с огромными вычислительными трудностями, т.к. для поиска такой подсистемы, строго говоря, необходимо осуществлять прямой перебор всех возможных комбинаций соотношений, входящих в эти подсистемы. Поэтому актуальна также задача построения метода и алгоритма, обеспечивающего получение оценок параметров моделей цветовой коррекции, обладающих свойствами согласованных оценок, но отличающихся приемлемой вычислительной сложностью.

Как уже отмечалось выше, использование методов статистического оценивания при малом числе наблюдений оказывается малообоснованным. Ниже излагается метод идентификации, основанный на целенаправленном поиске переменных, свободных от шума.

Идея метода состоит в том, чтобы для исходной системы построить множество оценок на всех возможных подсистемах небольшой размерности и отобрать среди них подмножество наиболее согласованных между собой по критерию взаимной близости. На этом подмножестве затем строится искомая точечная оценка. Успех в данном случае зависит от того, насколько оправданы предположения о существовании некоторого подмножества наблюдений, почти свободных от шума. Указанные предположения более реалистичны и менее обременительны, чем те, которые обычно используются в теории статистического оценивания. Развиваемый подход не требует выполнения требования статистической устойчивости, которое все равно не выполняется при малом числе наблюдений. В данном случае используются следующие предположения.

1. Матрица S и вектор $\mathbf{z}$ фиксированы, т. е. $s_{i,j}, z_i$, $i=\overline{1,N}$, $j=\overline{1,M}$ известны в результате измерений на одной реализации.
2. Число наблюдений мало, поэтому имеет место неопределенность свойств обусловленности матрицы $\mathbf{S}$ и статистических характеристик вектора ξ.

3. Норма вектора ошибок $\boldsymbol{\Xi} = \left[\xi_1, \xi_2, ..., \xi_N\right]^T$ ограничена, а его направление в шаре

$$\boldsymbol{\Xi} = \left\{\boldsymbol{\xi} : \left(\boldsymbol{\xi}^T \boldsymbol{\xi}\right)^{1/2} = \left\|\boldsymbol{\xi}\right\|_2 \le R_\xi = \text{Const}\right\} \quad (15)$$

является произвольным и неопределенным.

4. В исходных данных, несмотря на возможные ошибки, все же содержится достаточное число наблюдений (подсистема, наиболее свободная от шума), по которым оценки $\hat{\mathbf{c}}$ могут быть вычислены с требуемой точностью.
5. Существует соответствующая уравнению (13) точная модель:

$$\mathbf{Z}^* = \mathbf{S}\mathbf{c}, \quad (16)$$

где $\mathbf{Z}^* = \mathbf{Z} - \boldsymbol{\Xi}$.

Конкретизируем сделанные предположения. Матрица $\mathbf{S}$, вообще говоря, случайна, однако в каждом эпизоде идентификации она формируется по доступным для непосредственного измерения (хотя и содержащим случайные ошибки) сигналам. Поэтому она считается известной точно, а содержащиеся в ней ошибки представляются в виде реализации случайного вектора ошибок $\boldsymbol{\Xi}$, длина и направление которого в пределах шара (15) являются неопределенными.

В дополнение к указанным может использоваться также предположение об ограничениях параметров модели в виде неравенств. Эти ограничения задаются на основе априорной информации о допустимых диапазонах оцениваемых характеристик. Аналогичные ограничения используются в традиционной постановке задачи оценивания параметров линейной регрессии и в данном случае не являются принципиальными.

Число наблюдений считается малым, если число степеней свободы $\left(N - M\right)$ превышает число оцениваемых параметров M не более чем в 2-2,5 раза. Что касается *предположения 4*, конечно, невозможно построить точную систему (16) из (13), т.к. не известен вектор ошибок $\boldsymbol{\Xi}$. Однако, опираясь на это

предположение, можно ставить задачу отыскания подсистемы, наиболее свободной от шума. Подчеркнем, что в данном случае решается задача идентификации по одному малому набору данных в рамках указанных выше *предположений 1-5*. Поэтому постановка задачи, по существу, является детерминированной. Традиционно применяемые в статистической теории оценивания вероятностные модели в данном случае не используются, поскольку на одной короткой реализации прогноз поведения ошибок оценивания, исходя из распределений ошибок измерений, не надежен.

В соответствии с изложенным принципом отбора данных общая схема построения согласованных оценок в общем случае представляется следующим образом. Из системы (13) сформируем множество так называемых подсистем нижнего уровня с помощью весовых матриц диагонального вида – $\mathbf{G}_k = \mathrm{diag}(g_{k,1},...,g_{k,N})$, $k = 1,2,\ldots$. Элементы этих матриц могут быть только нулями и единицами. Ненулевые элементы задаются для различных сочетаний из S_k индексов, так что $\mathrm{rank}\,\mathbf{G}_k = S_k$ для всех k. В результате получаем множество подсистем вида:

$$\mathbf{Z}_k = \mathbf{S}_k\mathbf{c}_k + \boldsymbol{\Xi}_k, \quad k = 1,2,\ldots, \qquad (17)$$

где

$$\mathbf{Z}_k = \mathbf{G}_k\mathbf{Z},\ \mathbf{S}_k = \mathbf{G}_k\mathbf{S},\ \boldsymbol{\Xi}_k = \mathbf{G}_k\boldsymbol{\Xi},$$

$$\dim\left[R(\mathbf{S}_k)\right] = \mathrm{rank}\,\mathbf{G}_k = \sum_{i=1}^{N} g_{k,i} = \|\mathbf{g}_k\|_2 = S_k,$$

где $R(\mathbf{S}_k)$ – пространство столбцов матрицы $\mathbf{S}_k$, а $\mathbf{g}_k$ – $N\times 1$-вектор: $\mathbf{G}_k = \mathbf{E}\mathbf{g}_k$.

Если размерность подсистем нижнего уровня фиксирована, т.е. $S_k = S$, то число подсистем нижнего уровня равно $C_N^S\ (S < N)$. Вычисляя для каждой из построенных таким образом подсистем МНК-оценку (14):

$$\hat{\mathbf{c}}_k = \left[\mathbf{S}^T\mathbf{G}_k\mathbf{S}\right]^{-1}\mathbf{S}^T\mathbf{G}_k\mathbf{Z},$$

получаем множество Ω всех возможных оценок на подсистемах нижнего уровня размерности S:

$$\Omega = \left\{ \hat{\mathbf{c}}_k = \left[\mathbf{S}^T \mathbf{G}_k \mathbf{S} \right]^{-1} \mathbf{S}^T \mathbf{G}_k \mathbf{Z} \middle| \forall \mathbf{G}_k : \ \left\| \mathbf{g}_k \right\| = S \right\}, \quad \left| \Omega \right| = C_N^S .$$

Аналогичным образом (из нулей и единиц) строится множество диагональных $N \times N$ весовых матриц $\mathbf{H}_l$ $\operatorname{rank} \mathbf{H}_l = P \left(S < P < N \right)$:

$$\mathbf{H}_l = \operatorname{diag}(h_{l,1}, ..., h_{l,N}),$$

с использованием которых формируются так называемые подсистемы верхнего уровня:

$$\tilde{\mathbf{Z}}_l = \tilde{\mathbf{S}}_l \mathbf{c}_l + \tilde{\boldsymbol{\Xi}}_l \ \tilde{\mathbf{y}}_l = \tilde{\mathbf{X}}_l \mathbf{c}_l + \tilde{\boldsymbol{\xi}}_l ,$$

где

$$\tilde{\mathbf{Z}}_l = \mathbf{H}_k \mathbf{Z}, \quad \tilde{\mathbf{S}}_k = \mathbf{H}_l \mathbf{S}, \quad \tilde{\boldsymbol{\Xi}}_l = \mathbf{H}_l \boldsymbol{\Xi}, \quad l = \overline{1, L}, \quad L = C_N^P .$$

Ясно, что каждой такой подсистеме будет соответствовать свое подмножество подсистем нижнего уровня и, соответственно, подмножество промежуточных оценок:

$$\Theta_l = \left\{ \hat{\mathbf{c}}_k \in \Omega \middle| \forall k : \mathbf{g}_k \mathbf{h}_l = \left\| \mathbf{g}_k \right\| = S \right\}, \quad \left| \Theta_l \right| = C_S^P, \quad \forall l = \overline{1, C_N^S} ,$$

где h_l – $N \times 1$-вектор: $\mathbf{H}_l = \mathbf{E}\mathbf{h}_l$. В общем случае размерность подсистем нижнего уровня может быть в пределах $M \le P < N$. Далее рассматривается только случай $M = P$, т.е. $\left| \Theta_l \right| = C_S^M$, $\forall l = \overline{1, C_N^S}$.

С использованием введенных подсистем принцип согласованности формулируется следующим образом. Пусть для каждой i-й подсистемы нижнего уровня, сформированной из l-й подсистемы верхнего уровня, вычислен вектор оценок $\hat{\mathbf{c}}_{l,i}$. Пусть также задана функция, характеризующая взаимную близость множества решений $\hat{\mathbf{c}}_{l,i}$ на каждой подсистеме верхнего уровня:

$$W[l] = \sum_{\substack{i,j=1, \\ i \ne j}}^{K_l} \rho\left(\hat{\boldsymbol{c}}_{l,i}, \hat{\boldsymbol{c}}_{l,j} \right), \ l = \overline{1, L}, \quad (18)$$

где $\rho\left(\hat{\boldsymbol{c}}_{l,i}, \hat{\boldsymbol{c}}_{l,j} \right)$ – расстояние между полученными на l-й подсистеме верхнего уровня векторами $\hat{\mathbf{c}}_{l,i}$ и $\hat{\mathbf{c}}_{l,j}$, взятыми попарно во всех возможных сочетаниях

$C_{K_l}^2$. Искомая точечная оценка $\hat{\boldsymbol{c}}$ ищется по тому же или иному правилу на подсистеме с номером l^*:

$$W\left[l^*\right] = \min_{\forall l} W\left[l\right], \quad l = \overline{1, L}. \quad (19)$$

В рамках указанной постановки могут строиться различные алгоритмы, отличающиеся мерой близости $\rho\left(\hat{\boldsymbol{c}}_{l,i}, \hat{\boldsymbol{c}}_{l,j}\right)$. В данном случае следуя работе [32] будем использовать частный случай критерия (18), в котором в качестве функции взаимной парной близости используется сумма квадратов евклидовых норм разности всех возможных пар векторов оценок:

$$W_l\left[l\right] = \sum_{\substack{i,\, j=1,\\ i \neq j}}^{K_l} \left\| \hat{\boldsymbol{c}}_{l,i} - \hat{\boldsymbol{c}}_{l,j} \right\|_E^2, \; l = \overline{1, L}, \quad (20)$$

где $\hat{\boldsymbol{c}}_{l,i}$, $\hat{\boldsymbol{c}}_{l,j}$, $i = \overline{1, K_l}$, $j = \overline{1, K_l}$ оценки, полученные на подсистемах нижнего уровня каждой (l-й) подсистемы верхнего уровня.

Основная проблема построения согласованных оценок состоит в том, что для их вычисления в соответствии с критерием (19) необходимо осуществлять полный перебор возможных комбинаций оценок на подсистемах нижнего уровня для всех подсистем верхнего уровня. Даже при сравнительно небольшой размерности задачи такой перебор становится неприемлемым для обычных вычислительных средств. Далее рассматривается метод формирования наиболее согласованных множеств наблюдений, в котором удается избежать полного перебора. При этом не строятся подсистемы верхнего уровня, а для остановки процедуры задают лишь допустимое число используемых оценок, взятых из пула.

Для реализации предлагаемой схемы формирования множества согласованных оценок задается начальная оценка, обозначаемая далее как $\hat{\mathbf{c}}_1$: $\hat{\mathbf{c}}_1 \in \Phi_1 \subset \Omega$, вычисленная на некоторой подсистеме нижнего уровня. Далее множество согласованных оценок формируется путем последовательного присоединения к созданному на очередном шаге промежуточному множеству

$\Phi_q, q=1,2,...$ оценки $\hat{\mathbf{c}}_{q+1}$ из дополнения к формируемому множеству $\left(\hat{\mathbf{c}}_{q+1} \notin \Phi_q\right)$, удовлетворяющей условию

$$\hat{\mathbf{c}}_{q+1}: \quad w\left[\hat{\mathbf{c}}_{q+1}\right] = \min_{\forall \hat{\mathbf{c}}_{q+1} \notin \Phi_q} \sum_{i=1}^{q} \left\| \mathbf{c}_i - \mathbf{c}_{q+1} \right\|_E^2 . \quad (21)$$

В качестве критерия остановки задается допустимое число присоединяемых оценок.

Утверждение 1:

Пусть $\hat{\mathbf{c}}_1$ – любая оценка из множества согласованных оценок: $\hat{\mathbf{c}}_1 \in \Phi_1 \subset \Omega$, удовлетворяющих критерию (20). Тогда множество согласованных оценок, сформированное по правилу (21) и содержащее C_n^M оценок, совпадает с множеством согласованных оценок для подсистемы верхнего уровня, удовлетворяющей критерию (21) и построенной методом полного перебора.

Доказательство.

Для всего множества последовательно присоединяемых векторов оценок выпишем значения сумм (21):

$$w\left[\hat{\mathbf{c}}_2\right] = \min_{\forall \hat{\mathbf{c}}_i \notin \Phi_1} \left\| \mathbf{c}_i - \mathbf{c}_1 \right\|_E^2, \quad (22)$$

$$w\left[\hat{\mathbf{c}}_3\right] = \min_{\forall \hat{\mathbf{c}}_i \notin \Phi_2} \left\{ \left\| \mathbf{c}_i - \mathbf{c}_1 \right\|_E^2 + \left\| \mathbf{c}_i - \mathbf{c}_2 \right\|_E^2 \right\}, \quad (23)$$

…

$$w\left[\hat{\mathbf{c}}_{C_S^M}\right] = \min_{\forall \hat{\mathbf{c}}_i \notin \Phi_{C_S^M - 1}} \sum_{j=1}^{C_S^M - 1} \left\| \hat{\mathbf{c}}_i - \hat{\mathbf{c}}_j \right\|_E^2$$

Если число последовательно присоединяемых по указанной схеме векторов оценок равно C_n^M, соответствующая их объединению сумма принимает вид:

$$W\left[C_S^M\right] = \sum_{q=2}^{C_S^M} w\left[\hat{\mathbf{c}}_q\right] = \sum_{q=2}^{C_S^M} \left\{ \min_{\forall \hat{\mathbf{c}}_i \notin \Phi_q} \sum_{j=1}^{q} \left\| \hat{\mathbf{c}}_i - \hat{\mathbf{c}}_j \right\|_E^2 \right\}.$$

Нетрудно заметить, что $W\left[C_S^M\right]$ является минимальным значением суммы квадратов разности всех возможных парных сочетаний множества C_n^M

оценок, для подсистемы, содержащей n уравнений. Следовательно $W\left[C_n^M\right]$ суть функция взаимной близости оценок для подсистемы верхнего уровня, удовлетворяющей критерию (21), а множество входящих в эти частные суммы векторов оценок совпадает с множеством согласованных оценок, сформированным путем полного перебора всех подсистем верхнего уровня. ■

Приведенное утверждение говорит о том, множества согласованных оценок на подсистеме верхнего уровня одинаковой размерности, сформированные методом полного перебора и с использованием последовательной схемы отбора совпадают, если в результате реализации последовательной схемы формирования множества согласованных оценок отобраны все возможные их варианты. Обычно число согласованных оценок в последовательной схеме отбора оказывается значительно меньше. Если число M является делителем числа P , минимально возможное число таких оценок равно P/M. Тем не менее, даже в этом последнем случае сформированная подсистема верхнего уровня размерности P как правило совпадает с подсистемой той же размерности, но сформированной по методу полного перебора. Указанное свойство имеет место только в том случае, если начальная оценка $\hat{\mathbf{c}}_1$ принадлежит множеству согласованных оценок: $\hat{\mathbf{c}}_1 \in \Phi_1 \subset \Omega$, удовлетворяющих критерию (21). Поэтому представляет интерес установить условия, при которых это имеет место.

Пусть $\hat{\mathbf{c}}_k, k = \overline{1, C_N^M}$ – полное множество оценок, полученных на подсистемах нижнего уровня, а

$$\Delta\hat{\mathbf{c}}_k = \left[\mathbf{S}^T\mathbf{G}_k\mathbf{S}\right]^{-1}\mathbf{S}^T\mathbf{G}_k\boldsymbol{\Xi}$$

– соответствующее множество ошибок оценивания. Предположим, что в исходной системе имеется подсистема почти свободная от шума (назовем ее точной), а также содержатся зашумленные данные. Зададим некоторую норму и построим вариационный ряд норм ошибок оценивания. Если число наблюдений, входящих в точную подсистему таково, что число построенных на ней подсистем нижнего уровня больше, чем число подсистем нижнего уровня, со-

держащих зашумленные наблюдения, то средний член этого вариационного ряда будет принадлежать множеству согласованных оценок.

Представляет интерес установить сколько наблюдений должно входить в точную подсистему. В соответствии со сказанным выше для того, чтобы оценка $\hat{\mathbf{c}}_1$, полученная на некоторой подсистеме нижнего уровня и соответствующая среднему члену вариационного ряда, составленного из норм векторов ошибок оценивания, являлась также одной из согласованных оценок, необходимо, чтобы существовало n наблюдений, принадлежащих точной подсистеме, для которых

$$2C_n^M > C_N^M .$$

Из этого неравенства нетрудно получить условие существования подсистемы почти свободной от шума, включающей множество $]n[$ наблюдений, где $]n[$ ближайшее (в сторону увеличения) целое число, при выполнении которого оценка $\hat{\mathbf{c}}_1$ принадлежит множеству согласованных оценок:

$$2\big(]n[-M+1\big)...\big(]n[-1\big)]n[>\big(N-M+1\big)...\big(N-1\big)N . \quad (24)$$

Интересно, что при M=1, как и следовало ожидать, условие (2.18) сводится к требованию

$$]n[>\frac{N}{2} .$$

При увеличении числа оцениваемых параметров M множество наблюдений n всегда будет меньше половины всех наблюдений.

2. Параллельная реализация процедуры согласованной идентификации на суперкомпьютере

Ввиду высокой вычислительной сложности алгоритма согласованной идентификации представляет практическую ценность параллельная реализация с использованием технологии MPI.

Распределение вычислений на *n* процессов проводится следующим образом. На этапе инициализации каждый процесс получает исходную матрицу задачи **X** и вектор **y**. Корневой процесс генерирует *n* последовательностей бинарных векторов N×1 с нормой Хэмминга равной S и рассылает их на *n* процессов. В случае гомогенной структуры вычислительной системы каждому процессу посылаются пакеты одинаковой длины. Параллельные процессы выполняют расчет оценок ля каждого бинарного вектора из пакета, результат расчетов пересылают корневому процессу.

Если среда гетерогенная, то корневой процесс сначала рассылает пакеты одинаковой длины, после прихода результатов расчета от процесса ему посылается больший или меньший пакет, пропорционально скорости, с которой был обработан текущий пакет. Такой способ позволяет так же учесть динамическое изменение вычислительных мощностей системы при немонопольном ее использовании.

Перебор b_i – векторов, определяющих подсистемы, в порядке возрастания в двоичном коде или коде Грея не может обеспечить постоянство нормы Хэмминга для всех вариантов. Это приводит к появлению большого количества “холостых” итераций перебора. В следующем разделе приводится алгоритм, позволяющий последовательно получать бинарные вектора с заданной длиной и постоянной нормой Хэмминга.

Второй этап согласованного оценивания заключается непосредственно в вычислении значения критерия взаимной близости на каждой подсистеме верхнего уровня и определении наиболее согласованной подсистемы. В случае небольшого количества подсистем верхнего уровня этот этап может проводиться на корневом процессе по мере поступления от параллельных процессов данных о значениях оценок на подсистемах нижнего уровня.

Параллельная реализация требует наличия у каждого параллельного процесса данных обо всем множестве промежуточных оценок. Эти данные может рассылать всем процессам корневой процесс. Однако, в целях сокращения объема межпроцессного обмена в системах с разделенной памятью

каждый процесс по мере окончания расчета может самостоятельно рассылать результаты остальным процессам.

Когда процессы заканчивают расчет промежуточных оценок, корневой процесс рассылает последовательности бинарных векторов кодирующих подсистемы верхнего уровня. Параллельные процессы выполняют вычисление критерия взаимной близости, и определяется наиболее согласованная подсистема верхнего уровня, по которой вычисляется оценка. Можно представить параллельную реализацию согласованного оценивания в виде следующей схемы.

Master процесс:

- Генерация векторов описывающих подсистемы для вычисления оценок.
- Рассылка наборов векторов slave процессам при помощи MPI_Send.
- Сбор оценок от slave процессов при помощи MPI_Recv.
- Вычисление критерия взаимной близости и определение окончательной оценки.

Slave процесс:

- Ожидание получения наборов векторов от master процесса при помощи MPI_Recv.
- Цикл по полученному набору векторов для вычисления МНК оценок.
- Передача полученных оценок master процессу при помощи MPI_Send.

Особый интерес представляет процесс генерации двоичных векторов**.** Перебор b_i в порядке возрастания в двоичном коде или коде Грея не может обеспечить постоянство нормы Хэмминга для всех вариантов. Это приводит к появлению большого количества “холостых” итераций перебора.

Осуществить перебор b_i, где i изменяется от 0 до $I = C_N^k - 1$, с сохранением нормы Хэмминга позволяет алгоритм, состоящий из следующих шагов.

1. Пусть стартовым значением процедуры перебора будет $\mathbf{b}_0 = \underbrace{0\ldots0}_{N-k}\underbrace{1\ldots1}_{k}$.
2. На *i*-ом шаге выполняется следующая процедура. Пусть *l* наименьший разряд, для которого $\mathbf{b}_{i-1}^{l-1} = 1$ и $\mathbf{b}_{i-1}^{l} = 0$. Тогда $\mathbf{b}_i^j = \mathbf{b}_{i-1}^j$ при $j > l$, $\mathbf{b}_i^l = 1$ и $\mathbf{b}_i^{l-1} = 0$. Если $\mathbf{b}_{i-1}^0 = 0$ и $l > 1$, тогда $\mathbf{b}_i^{l-1-j} = \mathbf{b}_{i-1}^j$ при $j < l$ *-1*. Если необходимого *l* не найдено, значит, перебор завершен и получен вектор $\mathbf{b}_I = \underbrace{1\ldots1}_{k}\underbrace{0\ldots0}_{N-k}$.
3. Если не получено b_I переход на шаг 2.

в таблице 3.1 приведена последовательность b_i для $N = 5$ и k=3, $I = C_6^3 - 1 = 19$.

Таблица 3.1 Формирование двоичного представления переопределенных систем

I	**Bi**	**i**	**b_i**	**i**	**b_i**	**i**	**b_i**
0	**000111**	**5**	**010101**	**10**	**100011**	**15**	**101100**
1	**001011**	**6**	**010110**	**11**	**100101**	**16**	**110001**
2	**001101**	**7**	**011001**	**12**	**100110**	**17**	**110010**
3	**001110**	**8**	**011010**	**13**	**101001**	**18**	**110100**
4	**010011**	**9**	**011100**	**14**	**101010**	**19**	**111000**

Задача оценивания выполняется для каждой переопределенной системы, определяемой вектором b_i. Оценивание для нескольких значений размерности переопределенных систем можно выполнить для каждой размерности при помощи описанной процедуры.

3. Пример программного кода MATLAB для цветовой коррекции.

```
function MBEnh_0_1
clear all;
image_filename = 'MB_1.tif';
image_filetype = 'TIFF';
mask_filename = 'Mask_1.tif';
mask_filetype = 'TIFF';
data_name = 'data.mat';
```

```
%сканер точек с маски
 data_name = scan_mask_2(mask_filename, mask_filetype);
%/сканер точек с маски
load(data_name);
image = imread(image_filename,image_filetype);
x_points = [flare_borders(:,1);sample_points(:,1)];
y_points = [flare_borders(:,2);sample_points(:,2)];
C = makecform('srgb2lab');
sample_colors = applycform(sample_colors,C);
diff_colors = zeros(1,size(sample_points,1));
for ii=1:size(sample_points,1)
    mean_color = squeeze(mean(mean(image(sample_points(ii,1)-
1:sample_points(ii,1)-
1+2,sample_points(ii,2):sample_points(ii,2)+3,:))));
    diff_colors(ii) = log(abs(mean_color(1)-
double(sample_colors(ii,1))));
    a_colors_delta(ii) = double(sample_colors(ii,2))-
mean_color(2);
    b_colors_delta(ii) = double(sample_colors(ii,3))-
mean_color(3);
end
z_points = [ones(size(flare_borders,1),1); diff_colors'];
options = optimset('MaxFunEvals',5000,'MaxIter', 10000);
C = lsqcurve-
fit(@myfun,[0,2500,5000,1,1,200],[double(x_points)';double(y_poi
nts)'],double(z_points'),[-
pi/2,0,0,0,0,0],[pi/2,10000,10000,1,1,10],options);
vec = (C(4)*(cos(C(1))*(double(x_points)'-C(2))-
sin(C(1))*(double(y_points)'-
C(3))).^2+C(5)*(sin(C(1))*(double(x_points)'-
C(2))+cos(C(1))*(double(y_points)'-C(3))).^2).^0.5;
myfun3 = @(p,x) p(1)*(x-p(3))./(1+p(2)*abs(x-p(3)))+p(4);
C_1 = lsqcurvefit(myfun3,[-
100,1,0,C(6)],vec,exp(double(z_points')),[-
1000,0,0,0],[0,1000,1,100],options);
%правим светлость
[X,Y] = meshgrid(1:2868,1:4774);
vec = (C(4)*(cos(C(1))*(X-C(2))-sin(C(1))*(Y-
C(3))).^2+C(5)*(sin(C(1))*(X-C(2))+cos(C(1))*(Y-C(3))).^2).^0.5;
Z = C_1(1)*(vec-C_1(3))./(1+C_1(2)*abs(vec-C_1(3)))+C_1(4);
corr_image = uint8(1/2*(sign(Z) + 1).*Z)';
```

```
% Z = -C(2)*(cos(C(1))*(X-C(4))-sin(C(1))*(Y-C(5))).^2-
C(3)*(sin(C(1))*(X-C(4))+cos(C(1))*(Y-C(5))).^2+C(6);
% corr_image = uint8(1/2*(sign(Z) + 1).*exp(Z))';
%/правим светлость
%правим цвета
a_colors = sample_colors(:,2);
b_colors = sample_colors(:,3);
z_max = double(max(max(corr_image)));
z_min = double(min(min(corr_image)));
percent = zeros(size(sample_points,1),1);
for ii=1:size(sample_points,1)
percent(ii) =
corr_image(sample_points(ii,1),sample_points(ii,2));
end
percent = (double(percent)-z_min)/(z_max-z_min);
a_colors_delta = double(a_colors_delta')./percent;
b_colors_delta = double(b_colors_delta')./percent;
a_min = min(min(image(:,:,2)));
a_max = max(max(image(:,:,2)));
b_min = min(min(image(:,:,3)));
b_max = max(max(image(:,:,3)));
a_colors=double([a_colors; uint8(ones(1,1))*a_max;
uint8(ones(1,1))*a_min]);
a_colors_delta = double([a_colors_delta; zeros(2,1)]);
b_colors=double([b_colors; uint8(ones(1,1))*b_max;
uint8(ones(1,1))*b_min]);
b_colors_delta = double([b_colors_delta; zeros(2,1)]);
C_a = poly-
fit(double(a_colors),double(a_colors)+a_colors_delta,2);
C_b = poly-
fit(double(b_colors),double(b_colors)+b_colors_delta,2);
x = 0:1:255;
y_a = polyval(C_a,x);
y_b = polyval(C_b,x);
plot(x,y_a,x,y_b,a_colors,a_colors+a_colors_delta,'o',b_colors,b
_colors+b_colors_delta,'x');
corr_image_percent = double(corr_image)/double(z_max);
for ii = 1:size(corr_image,1)
    ii
    for jj = 1:size(corr_image,2)
        weight = corr_image_percent(ii,jj);
```

```
        if weight>0
            image(ii,jj,2) = double(image(ii,jj,2))*(1-weight) +
polyval(C_a,double(image(ii,jj,2)))*weight;
            image(ii,jj,3) = double(image(ii,jj,3))*(1-weight) +
polyval(C_b,double(image(ii,jj,3)))*weight;
        end
    end
end
%/правим цвета
image(:,:,1) = uint8(image(:,:,1)-corr_image);
imwrite(image,'4.tiff','tiff','ColorSpace','icclab');
return
end
function F = myfun(p,x)
F = -p(4)*(cos(p(1))*(x(1,:)-p(2))-sin(p(1))*(x(2,:)-p(3))).^2+-
p(5)*(sin(p(1))*(x(1,:)-p(2))+cos(p(1))*(x(2,:)-p(3))).^2+p(6);
%p(2)*(1-sqrt((x+p(1)).^2));
end
function F = myfun1(p,x)
    vec = (p(2)*(cos(p(1))*(x(1,:)-p(4))-
sin(p(1))*(x(2,:)+p(5))).^2+p(3)*(sin(p(1))*(x(1,:)-
p(4))+cos(p(1))*(x(2,:)-p(5))).^2).^0.5;
F = p(6)*(vec-p(7))./(1+p(8)*abs(vec-p(7)))+p(9);
end
```

4. Выполнение лабораторной работы.

4.1 Подготовка.

1. Изучите краткую теоретическую справку о методах идентификации.
2. Для выполнения работы создайте персональную директорию, которая должна иметь следующий адрес:

   D:\ИНФОРМАТИКА\[НОМЕР ГРУППЫ]\[ФАМИЛИИ СТУДЕНТОВ],

   где поля [Номер группы] и [Фамилии студентов] соответствуют номеру ВАШЕЙ группы и ВАШИМ фамилиям.

   В полученную директорию скопируйте ВСЕ файлы из директории преподавателя:

   D:\ИНФОРМАТИКА\ЛАБОРАТОРНАЯ РАБОТА\ПРОГРАММА

   и файл со своим вариантом задания из директории:

   D:\ИНФОРМАТИКА\ЛАБОРАТОРНАЯ РАБОТА\ВАРИАНТЫ

3.2 Ход выполнения лабораторной работы.

1. С помощью графического пакета Adobe Photoshop откройте файл INPUT.BMP содержащий корректируемое изображение, файл MASK.BMP, предназначенный для задания тестовых фрагментов, и файл ETALON.BMP

Рисунок 4 – Пример эталонного изображения

Рисунок 5 – Пример искаженного изображения

2. Проанализируйте полученное изображение. Определите цвета, которые не соответствуют цветам эталона. Определите фрагменты изображения, на которых не соответствующие цвета наиболее постоянны.

Рисунок 6 – Определение подходящих тестовых фрагментов

3. Используя инструменты «цветовая проба» и «кисть» нанесите на изображение MASK точку (размер кисти установите 5 пикселей) с координатами определенного на предыдущем шаге фрагмента. Цветом для данной точки служит цвет данного фрагмента на изображении ETALON. Задайте таким образом от 10 до 15 различных по цвету точек.

Рисунок 7 – соответствующие пары цветов

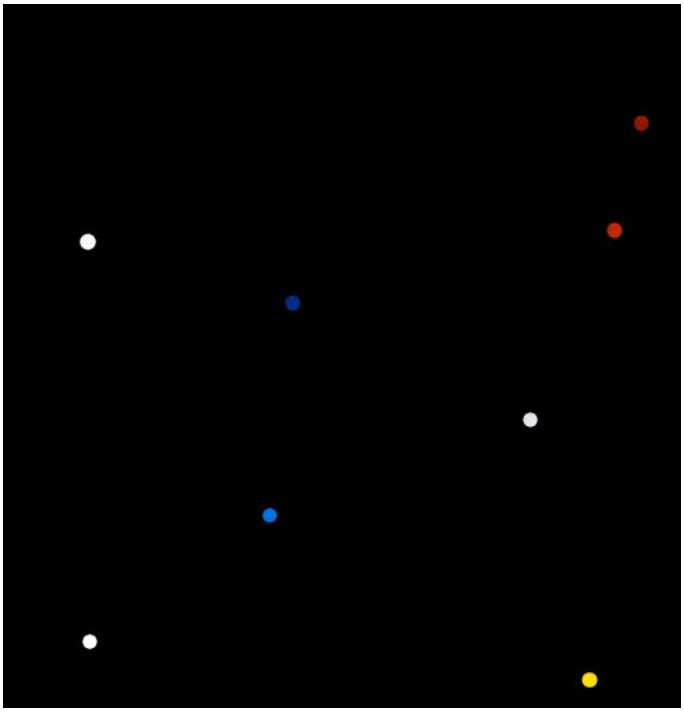

Рисунок 8 – Пример получаемой маски

4. Сохраните изменения в изображении MASK.
5. С помощью программного комплекса MATLAB запустите файл IDENT.M.
6. Сохраните изображения полученных в результате работы программы графиков.
7. Откройте результирующее изображение OUTPUT.BMP и сравните его с эталонным.

Рисунок 9 – Пример результирующего изображения

8. С использованием MS OFFICE WORD составьте отчет по проведенной лабораторной работе. В отчете следует проанализировать связь графиков кривых преобразования, исходного изображения и полученного результата. Если результирующее изображение существенно отличается от эталонного, объясните почему.
9. Сдайте отчет преподавателю и ответьте на контрольные вопросы.

4. Контрольные вопросы.

1. На каком принципе основано построение корректирующих кривых.
2. Какие ограничения накладываются на кривые преобразования? Почему?
3. Почему при малом количестве доступных измерений не следует использовать известные статистические методы оценивания?
4. Какое количество измерений относительно размерности системы можно считать «малым»?
5. В чем заключается основная идея согласованной идентификации?
6. Какими способами можно определять «желаемые» цвета для коррекции?
7. Какие типы процессов используются при распараллеливании задачи?

Список рекомендуемой литературы

1. **Бибиков, С.А.** Цветовая коррекция на основе идентификации моделей по тестовым фрагментам изображений [Текст] / С.А. Бибиков, В.А. Фурсов // Компьютерная оптика. - Самара-Москва, 2008. - Т.32 №3. - С.302-307.
2. **Мудров, В.И.** Методы обработки измерений [Текст] / В.И. Мудров, В.Л. Кушко. – М.: Сов. Радио, 1976. – 192с.
3. **Никоноров, А.В.** Распределенная вычислительная среда коррекции цветных изображений [Текст] / А.В. Никоноров, В.А. Фурсов // Труды XV Всероссийской научно-методической конференции "Телематика 2008", С.-Петербург, 23-26 июня, 2008. - С. 88-89.
4. **Попов, С.Б.** Кластерная технология формирования и параллельной фильтрации больших изображений [Текст] / С.Б. Попов, В.А. Сойфер, А.А. Тараканов, В.А. Фурсов // Компьютерная оптика. - Самара-Москва, 2002. - Т.23 №3. - С.75-83.
5. **Фурсов, В.А.** Проблемы вычисления оценок по малому числу наблюдений [Текст] / В.А. Фурсов // Лекция в трудах молодежной школы "Математическое моделирование 2001", Самара, 13-16 июня 2001. - С. 56-63.
6. **Шашлов, Б.А.** Цвет и цветовоспроизведение [Текст] / Б.А. Шашлов. - М.: Мир книги. - 1995. – 316 с.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

«УСТРАНЕНИЕ ТОЧЕЧНЫХ БЛИКОВ НА ЦИФРОВЫХ ИЗОБРАЖЕНИЯХ»

Составители:
С.А. Фурсов В.А., Никоноров А.В., Кузьмишина Т.М., Бибиков С.А.,
Якимов П.Ю., Минаев Е.Ю., Гайнулина Г.К.

Лабораторная работа предназначена для студентов специальности
“Технологии полиграфического производства” по курсу “Информатика”
и для специалистов, проходящих курсы повышения квалификации.
Одобрена решением кафедры общей информатики ГОУ ВПО
«Самарский государственный аэрокосмический университет имени
академика С.П.Королёва (национальный исследовательский университет)»,
протокол № 4 от 25 ноября 20010 г.

САМАРА 2010

Цель работы – изучение методов и алгоритмов устранения артефактов, возникающих при получении цифровой копии с живописного полотна. В лабораторной работе изучается метод устранения артефактов, возникающих при получении цифровой копии с живописного полотна.

1. Теоретические основы лабораторной работы.

Современная технология офсетной печати позволяет выпускать продукцию очень высокого качества. Возможна печать многокрасочных публикаций, информационной, рекламной продукции, а также репродукционное воспроизведение живописных полотен.

Необходимость проведения предпечатной обработки изображений при производстве цифровых репродукций произведений живописи связана со спецификой фотографирования оригиналов. В процессе фотографирования возникают следующие артефакты: множественные блики, затенения части холста рамкой и так называемые матовые блики. Артефакты обусловлены неровностями нанесения краски на холст и способом установки дополнительного освещения.

В лабораторной работе рассматривается алгоритм цифровой обработки изображений для удаления множественных бликов, которые возникают вследствие того, что отдельные малые участки слоя краски обладают таким углом наклона поверхности, что падающий свет источника отражается и регистрируется камерой в виде пятен. На цифровом изображении такие артефакты выглядят как области небольшого размера, обладающие выделяющейся на общем фоне яркостью. Пример изображения с артефактами представлен на Рисунке 1.

Коррекция множественных бликов выполняется в два этапа: поиск областей обладающих высокой относительно некой окрестности яркостью и замена цвета блика на цвета из его окрестности.

Если обработку изображения будет проводить специалист по компьютерной графике с использованием стандартных возможностей программы

Photoshop либо аналогичной, то для качественной ретуши изображения формата А3 ему понадобиться примерно 2-3 часа времени.

Был создан алгоритм, способный автоматически распознавать точечные артефакты и «заливать» их цветом пикселей в его окрестности. Алгоритм замещения бликов обладает высокой вычислительной сложностью, поэтому его реализация в виде последовательной программы требует значительных затрат времени для выполнения. Существенное ускорение данного алгоритма возможно путём его распараллеливания и применения NVIDIA CUDA (Compute Unified Device Architecture).

CUDA - это среда разработки на языке программирования Си, которая позволяет программистам и разработчикам писать программное обеспечение для решения сложных вычислительных задач за меньшее время благодаря многоядерной вычислительной мощности графических процессоров. Данная технология хорошо зарекомендовала себя за счет эффективного многократного ускорения множества алгоритмов и хорошо подходит для решения задач обработки изображений.

Рисунок 1 – Пример цифровой копии с точечными бликами

Постановка задачи

В лабораторной работе решается задача идентификации и исследования модели артефактов на цветном изображении. Эта задача имеет большое значение для процесса цветовоспроизведения в целом. Рассмотрим основные термины и модели, используемые в работе.

В настоящей работе изображение будет рассматриваться как двумерная функция вида $f(x,y)$, где x и y - координаты элемента изображения на плоскости, и значение f которой в любой точке, задаваемой парой координат (x,y), называется интенсивностью или уровнем серого (яркость) изображения в этой точке. Если величины x, y и f принимают конечное число дискретных значений, то говорят о цифровом изображении. Цифровой обработкой изображений называется обработка цифровых изображений с помощью цифровых вычислительных машин (компьютеров). Отметим, что цифровое изображение состоит из конечного числа элементов, каждый из которых расположен в конкретном месте и принимает определенное значение. Эти элементы называются элементами изображения или пикселями. Чаще всего для элементов цифрового изображения используется термин «пиксель».

Значения функции яркости f(x,y) как правило, принимают дискретные значения в диапазоне от 0 до 255 (8-битное кодирование), иногда – от 0 до 65535 (16-ти битное кодирование). В настоящей работе будет использоваться нормированное значение яркости в диапазоне от 0 до 1.

В реальных задачах обработки цветных изображений каждому пикселю соответствует вектор из нескольких (как правило, трех) координат цвета. В настоящей работе цветные изображения представляются в пространстве RGB. Таким образом, мы будем говорить о трех функциях яркости – определенных для каждого пикселя $f_R(x,y)$, $f_G(x,y)$ и $f_B(x,y)$. Кроме того, будем рассматривать функцию общей яркости или светлоты $L(x,y)$, определяемую согласно

$$L(x,y) = 0,3 f_R(x,y) + 0,59 f_G(x,y) + 0,11 f_B(x,y). \quad (1.1)$$

Задача ретуши будет решаться в виде двух подзадач:

- *обнаружение точечных артефактов на изображении,*
- *собственно ретушь, т.е. замещение цвета пикселей в области блика на некоторый цвет, близкий к цвету пикселей окрестности блика.*

Пример решения этой задачи для нескольких артефактов приведен на рисунке 2.

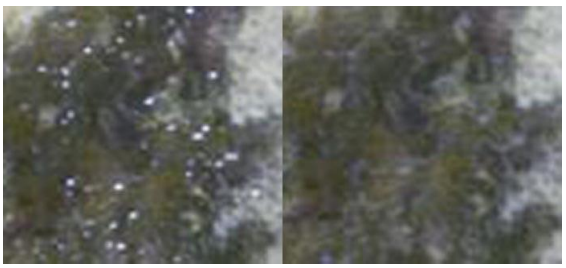

Рисунок 2 – Пример цифровой ретуши

Задача идентификации артефактов

Обнаружение артефактов будет проводиться нами по значениям светлоты, т.е. относительно функции (1.1), окончательная ретушь будет проводиться по всем трем каналам $f_R(x,y)$, $f_G(x,y)$ и $f_B(x,y)$ независимо.

Рассмотрим определение точечных артефактов, используемое в настоящей работе.

Под этим термином будем понимать относительно небольшие компактные области с высокой (по относительному или абсолютному значению) яркостью.

Понятия размеров и требуемой яркости области определяются на основе экспертных заключений цветокорректоров следующим образом. Область блика полагается вкладывающейся в квадрат 8x8 пикселей. Область требуемого размера является бликом, если средняя яркость превосходи яркость окружающей области более чем в 2,45 раза либо минимальная яркость более 0,7.

Такие параметры выбраны опытным путем, и позволяют автоматизировать процесс обнаружения артефактов, не повреждая мелкие смысловые фрагменты изображения.

Таким образом, при автоматизированном обнаружении артефактов в качестве отличительных характеристик должны использоваться высокая относительная или абсолютная яркость области в совокупности с компактностью и небольшой плотностью области как приведено на рисунке 3.

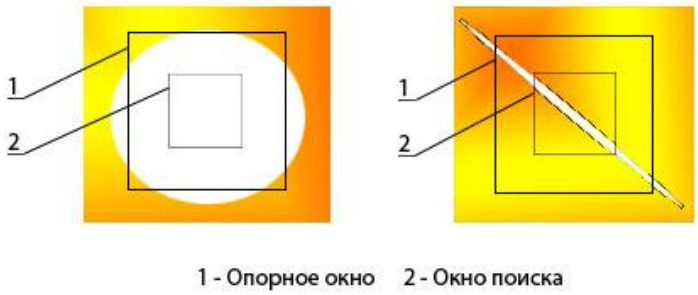

Рисунок 3 – Слева пример нарушение размера, справа нарушение компактности

Пусть W_1 – область изображения, покрытая бликом, W_0 некоторая окрестность области блика. Рассмотрим функцию принадлежности $I_0(x,y)$:

$$\left.\begin{array}{l} I_0(x,y)=1, \ \ (x,y)\in W_0 \\ I_0(x,y)=0, \ \ (x,y)\notin W_0 \end{array}\right\}. \qquad (1.2)$$

Тогда для блика W_0 должны выполняться следующие требования:

$$\max_{(x,y)\in W_1} f(x,y) \geq f_1, \qquad (1.3)$$

$$\frac{M_{(x,y)\in W_1} f(x,y)}{M_{(x,y)\in W_0} f(x,y)} \geq M_1, \qquad (1.4)$$

$$\sum I(x,y) \leq S_1, \qquad (1.5)$$

$$D(W_1) \leq D_1. \qquad (1.6)$$

Здесь $M(f(x,y))$ среднее значение функции яркости в области, $D(W_1)$ диаметр области, величина, характеризующая степень компактности области. Диаметр области можно определить, как максимальное евклидово расстояние между точками области:

$$D(W_1) = \max_{(x_1,y_1),(x_2,y_2)\in W_1} \left\| (x_1,y_1),(x_2,y_2) \right\|_2. \qquad (1.7)$$

Соотношения (1.3) и (1.4) отражают требования к максимальной и относительной яркости блика, а (1.5) и (1.6) – требование малой площади и компактности области. Константы f_1, M_1, S_1, D_1 являются определяющими для разрабатываемых алгоритмов и должны задаваться экспертами.

Задача замещения артефактов (ретушь)

Задача замещения цвета пикселей в области блика на некоторый цвет, близкий к цвету пикселей окрестности блика менее формализуема, чем задача обнаружения блика. Выбор цвета, который используется при замещении, производится на основе некоторого эвристического правила.

При обработке такого изображения оператором замещение выполняется следующим образом. На область блика клонируется некоторый участок окрестности, с наиболее характерным цветом и текстурой. Затем, выполняется размытие по Гауссу в области, чуть большей, чем область блика. Размытие позволяет избежать резких границ между ретушированным участком и его окрестностью.

Таким образом, можно сформировать следующую формальную последовательность этапов замещения блика:

1. Выбор набора пикселей из окрестности блика, копирование цвета этих пикселей в область блика:

$$W_1 = S(W_0), \qquad (1.8)$$

где S некоторый оператор выбора замещающего набора.

2. Расширение области блика:

$$\tilde{W}_1 = D(W_1), \qquad (1.9)$$

где D некоторый оператор расширения области, например, дилатация.

3. Сглаживающая фильтрация гауссовым фильтром в расширенной области

$$\tilde{W}_1 = F(\tilde{W}_1), \qquad (1.10)$$

где F оператор сглаживающего фильтра.

Соотношения (1.8)-(1.10) всего лишь формальная запись требуемых действий, каждый из этих этапов может быть реализован на основе различных алгоритмов. Выбор алгоритмов предлагаемых в настоящей работе был выполнен

на основе экспертной оценки качества получаемого в результате ретушированного изображения.

Алгоритм цифровой ретуши точечных артефактов

Шифрование

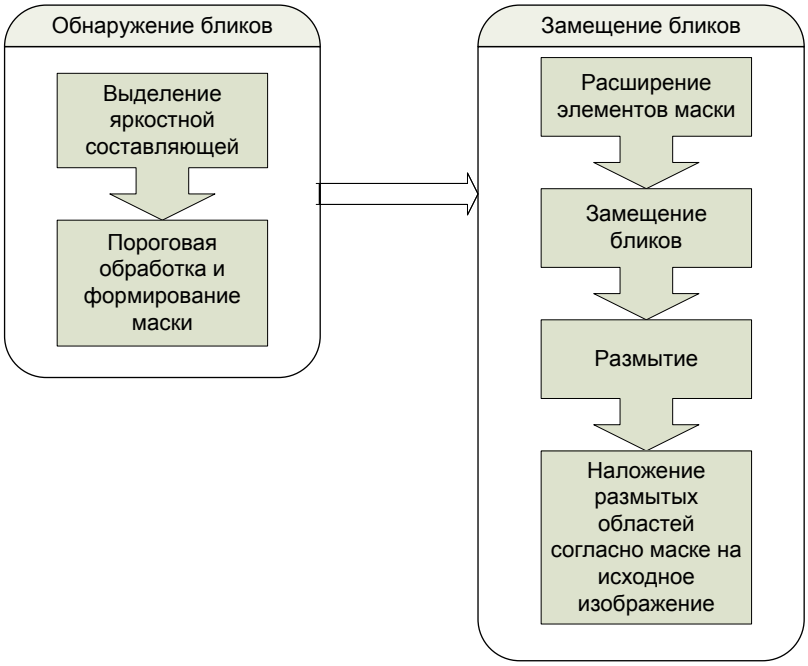

Рисунок 4 – Схема алгоритма цифровой ретуши точечных артефактов

На рисунке 4 приведена схема работы алгоритма, остановимся подробнее на каждом из этапов. Следуя соотношениям (1.3) и (1.4) для обнаружения блика необходимо сравнить яркость внутри некоторой небольшой области, проверяемой на наличие блика, с яркостью в ее окрестности. Таким образом, для обнаружения блика необходимо сравнивать соотношения яркости двух областей. В качестве предварительного шаг алгоритма для каждого пикселя изображения выполняется расчет функции яркости (1.1), дальнейшая работа алгоритма проводится по значениям этой функции. В качестве базового используется алгоритм адаптивного порога [1]. Для алгоритмов адаптивного порога характерна оконная обработка изображения. В предлагаемом алгоритме обнаружения артефактов используется два вложенных окна.

Согласно обозначениям (1.2), область, проверяемую на наличие блика, будем обозначать W_1. Представим эту область в виде квадратного окна со стороной h_{w1} и будем называть окном поиска. Окрестность окна поиска – область W_1, будем называть опорным окном. Окно содержащие в себе оба окна опорное и поиска, представим в виде квадратного окна со стороной h_w и обозначим W. Соотношение опорного окна, и окна поиска представлены на рисунке 3.

Таким образом, весь алгоритм обнаружения заключается в последовательном прохождении опорного и поискового окна по изображению. Для каждого положения она поиска в зависимости от соотношений яркостей опорного и поискового окон принимается решение, принадлежит ли центральный пиксель окна поиска блику или нет.

Для выбранного расположения опорного и поискового окон рассмотрим последовательность шагов алгоритма.

Первые несколько проверок на выполнение требований (1.3) и (1.4), предъявляемых к яркости блика.

1. Проверка выполнения соотношения (1.4), если не выполняется, то блик отсутствует:

2. Проверка соотношения (1.3), если не выполняется, то блик отсутствует:

Далее, в алгоритме, проверяется площадь предполагаемого блика и его компактность согласно (1.5) и (1.6).

3. Если окно W содержит большое число ярких пикселей, то блик отсутствует, а яркие пиксели являются смысловыми элементами изображения:

$$\left.\begin{array}{l} f(x^*, y^*) - \max\limits_{(x,y)\in W} f(x,y) \le s \\ \sum\limits_{(x^*,y^*)\in W} I(x^*, y^*) \ge N \end{array}\right\}, \qquad (1.11)$$

где N параметр, характеризующий количество “ярких” пикселей в окне, а s – допустимое отклонение яркости “ярких” пикселей от максимальной, I – индикаторная функция (1.2). Фактически, условие (1.11) отражает требование о компактности области блика (1.6).

Требование небольшой площади блика (1.5) является определяющим при выборе размера окна поиска – h_{w1}:

$$\sum_{(x,y)\in W_1} I(x^*, y^*) = S_1 = h_{w1}^2 \ . \qquad (2.24)$$

Таким образом, наличие блика подтверждается при выполнении условий (1.3) и (1.4) и не выполнении (1.11).

Действие алгоритма определяется следующим набором параметров. Размеры основного и опорного окна – h_{w0}, h_{w1}. Максимальная относительная и абсолютная яркость блика – M_1, f_1. Максимально допустимое количество ярких пикселей – N, допустимое отклонение яркости ярких пикселей от максимальной – s. Значения этих параметров могут быть либо определены постоянно в результате ряда экспериментов, либо задаваться оператором.

Многопоточная реализация алгоритма устранения точечных бликов с помощью технологии CUDA

Последовательный алгоритм, разработанный для решения задачи по обработке цифрового изображения, требует достаточно большое время на выполнение. Проблема заключается в том, что на практике обрабатываемые изображения имеют размеры не менее 15 мегапикселей. Вследствие чего, время обработки является сравнительно большим и занимает от 2 минут в зависимости от ресурсов компьютера и размера изображения. Именно с целью устранения этого недостатка и было разработано приложение с применением технологии NVIDIA CUDA.

Технология CUDA — это программно-аппаратная вычислительная архитектура NVIDIA, основанная на расширении языка Си, которая даёт возможность организации доступа к набору инструкций графического ускорителя и управления его памятью при организации параллельных вычислений. CUDA помогает реализовывать алгоритмы, выполнимые на графических процессорах видеоускорителей GeForce восьмого поколения и старше (серии GeForce 8, GeForce 9, GeForce 200), а также Quadro и Tesla.

Первым шагом при переносе существующего приложения на CUDA - это его профилирование и определение участков кода «тормозящих» работу. Если среди таких участков есть подходящие для быстрого параллельного исполнения, эти функции переносятся на Си расширения CUDA для выполнения на GPU. Программа компилируется при помощи поставляемого NVIDIA компилятора, который генерирует код и для CPU, и для GPU. При исполнении программы, центральный процессор выполняет свои порции кода, а GPU выполняет CUDA код с наиболее тяжелыми параллельными вычислениями. Эта часть, предназначенная для GPU, называется ядром (kernel). В ядре определяются операции, которые будут исполнены над данными.

Видеочип получает ядро и создает копии для каждого элемента данных. Эти копии называются потоками (thread). Поток содержит счётчик, регистры и

состояние. Для больших объёмов данных, таких как обработка изображений, запускаются миллионы потоков. Потоки выполняются группами по 32 штуки, называемыми warp. Warp назначается исполнение на определенных потоковых мультипроцессорах. Каждый мультипроцессор состоит из восьми ядер — потоковых процессоров, которые выполняют одну инструкцию MAD за один такт. Для исполнения одного 32-поточного warp требуется четыре такта работы мультипроцессора (имеется ввиду частота shader domain, которая равна 1.5 ГГц и выше).

Мультипроцессор не является традиционным многоядерным процессором, он отлично приспособлен для многопоточности, поддерживая до 32 warp единовременно. Каждый такт аппаратное обеспечение выбирает, какой из warp исполнять, и переключается от одного к другому без потерь в тактах. Если проводить аналогию с центральным процессором, это похоже на одновременное исполнение 32 программ и переключение между ними каждый такт без потерь на переключение контекста. Реально ядра CPU поддерживают единовременное выполнение одной программы и переключаются на другие с задержкой в сотни тактов.

Модель программирования в CUDA предполагает группирование потоков. Потоки объединяются в блоки потоков (thread block) — одномерные или двумерные сетки потоков, взаимодействующих между собой при помощи разделяемой памяти и точек синхронизации. Программа (ядро, kernel) исполняется над сеткой (grid) блоков потоков (thread blocks), см. рисунок ниже. Одновременно исполняется одна сетка. Каждый блок может быть одно-, двух- или трехмерным по форме, и может состоять из 512 потоков на текущем аппаратном обеспечении.

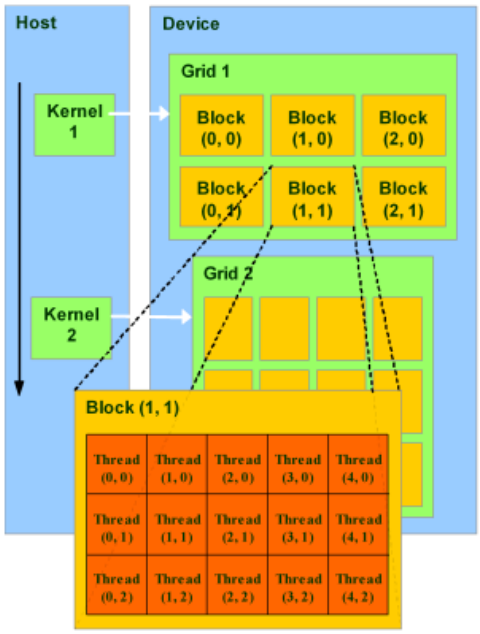

Рисунок 5 – Модель программирования CUDA

Блоки потоков выполняются в виде warp, размер которых — 32 потока. Это минимальный объём данных, которые могут обрабатываться в мультипроцессорах. И так как это не всегда удобно, CUDA позволяет работать и с блоками, содержащими от 64 до 512 потоков.

Группировка блоков в сетки позволяет уйти от ограничений и применить ядро к большему числу потоков за один вызов. Это помогает и при масштабировании. Если у GPU недостаточно ресурсов, он будет выполнять блоки последовательно. В обратном случае, блоки могут выполняться параллельно, что важно для оптимального распределения работы на видеочипах разного уровня, начиная от мобильных и интегрированных.

Модель памяти в CUDA отличается возможностью побайтной адресации, поддержкой как gather, так и scatter. Доступно довольно большое количество регистров на каждый потоковый процессор, до 1024 штук. Доступ к ним очень быстрый, хранить в них можно 32-битные целые или числа с плавающей точкой.

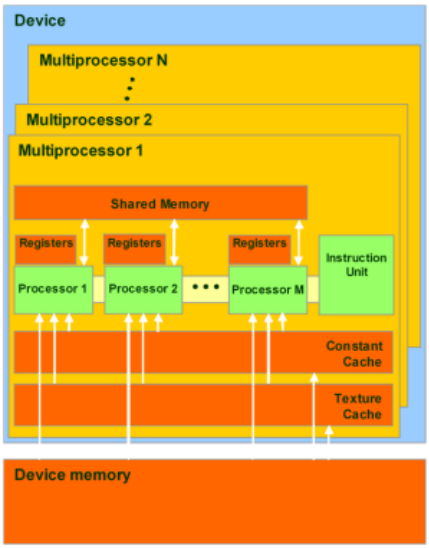

Рисунок 6 – Модель памяти CUDA

Глобальная память — самый большой объём памяти, доступный для всех мультипроцессоров на видеочипе, размер составляет от 256 мегабайт до 1.5 гигабайт на текущих решениях (и до 4 Гбайт на Tesla). Обладает высокой пропускной способностью, более 100 гигабайт/с для топовых решений NVIDIA, но очень большими задержками в несколько сот тактов. Не кэшируется, поддерживает обобщённые инструкции load и store, и обычные указатели на память.

Локальная память — это небольшой объём памяти, к которому имеет доступ только один потоковый процессор. Она относительно медленная — такая же, как и глобальная.

Разделяемая память — это 16-килобайтный (в видеочипах нынешней архитектуры) блок памяти с общим доступом для всех потоковых процессоров в мультипроцессоре. Эта память весьма быстрая, такая же, как регистры. Она обеспечивает взаимодействие потоков, управляется разработчиком напрямую и имеет низкие задержки. Преимущества разделяемой памяти: использование в виде управляемого программистом кэша первого уровня, снижение задержек при доступе исполнительных блоков (ALU) к данным, сокращение количества обращений к глобальной памяти.

Память констант — область памяти объемом 64 килобайта (то же — для нынешних GPU), доступная только для чтения всеми мультипроцессорами. Она

кэшируется по 8 килобайт на каждый мультипроцессор. Довольно медленная — задержка в несколько сот тактов при отсутствии нужных данных в кэше.

Текстурная память — блок памяти, доступный для чтения всеми мультипроцессорами. Выборка данных осуществляется при помощи текстурных блоков видеочипа, поэтому предоставляются возможности линейной интерполяции данных без дополнительных затрат. Кэшируется по 8 килобайт на каждый мультипроцессор. Медленная, как глобальная — сотни тактов задержки при отсутствии данных в кэше.

CUDA предполагает специальный подход к разработке, не совсем такой, как принят в программах для CPU. Нужно помнить о разных типах памяти, о том, что локальная и глобальная память не кэшируется и задержки при доступе к ней гораздо выше, чем у регистровой памяти, так как она физически находится в отдельных микросхемах.

Вид программного комплекса «Блик»и результаты обработки

На Рисунках 7-9 изображен общи1 вид программного комплекса для устранения точечных артефактов «Блик».

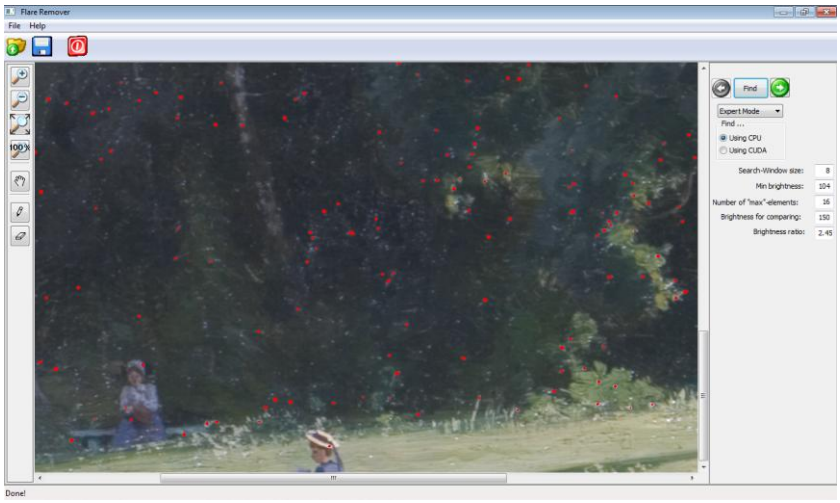

Рисунок 7 - Вид GUI программного комплекса «Блик».

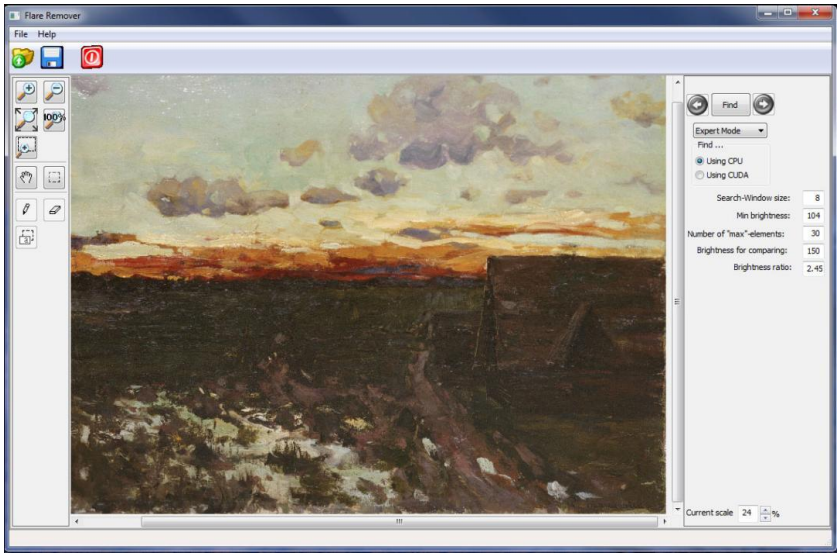

Рисунок 8 - Вид GUI программного комплекса «Блик».

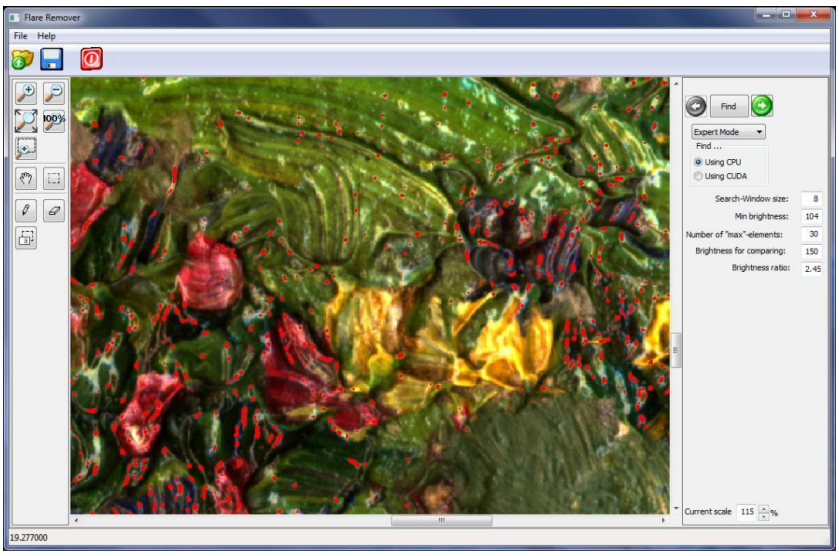

Рисунок 9 - Вид GUI программного комплекса «Блик»..

На Рисунках 10-13 изображены примеры работы реализованного алгоритма.

Рисунок 10 – пример обработки изображения

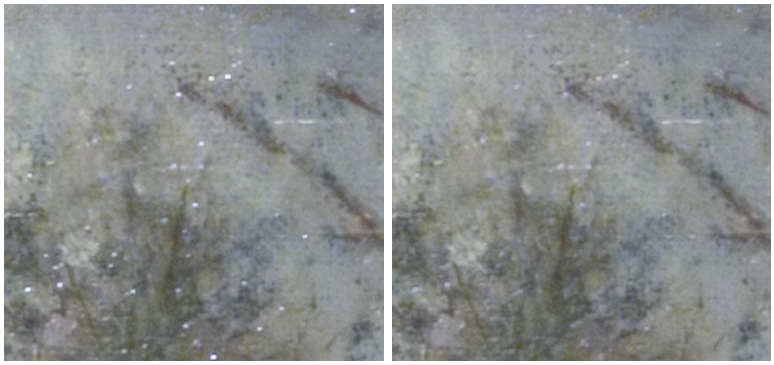

Рисунок 11 – Изображение до обработки (слева) и после фильтра (справа)

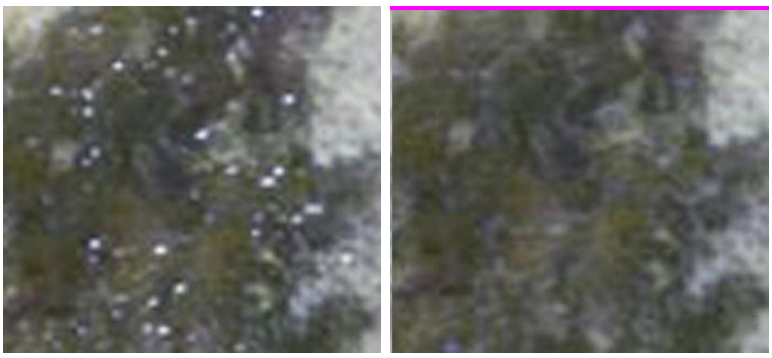

Рисунок 12 - Пример работы программной системы по устранению бликов на изображении «лес»: а) исходное, б) обработанное.

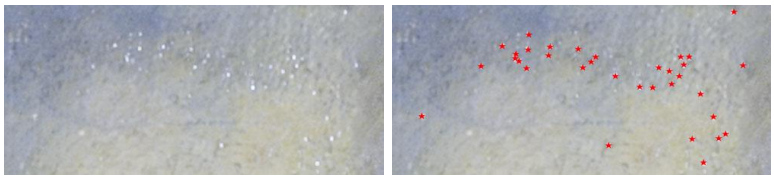

Рисунок 13 – Пример работы алгоритма идентификации

2. Выполнение лабораторной работы.

2.1. Общий план выполнения работы.

1. Изучить описание алгоритма устранения точечных артефактов.
2. Получить от преподавателя вариант задания с исходным изображением, содержащим блики.
3. Обработать исходное изображение «вручную», используя графический редактор (например, Adobe Photoshop).
4. Запустить и изучить программный комплекс «Блик».
5. Обработать исходное искаженное изображение с помощью программного комплекса «Блик».
6. Сравнить изображения, полученные «вручную» и с помощью программного комплекса «Блик»
7. Составить отчет о выполненной работе.
8. Сдать отчет преподавателю, ответить на контрольные вопросы, получить зачет по работе.

3. Контрольные вопросы.

1. Общее описание алгоритма.
2. Описание алгоритма идентификации точечных бликов.
3. Описание алгоритма ретуши точечных бликов.
4. Общее описание технологии CUDA.