Омский Государственный Университет им.Ф.М.Достоевского

На правах рукописи

Ларионов Игорь Борисович

Автоматическое восстановление поврежденных данных в графических файлах

05.13.18 – математическое моделирование, численные методы и комплексы программ

ДИССЕРТАЦИЯ

на соискание ученой степени

кандидата физико-математических наук

Научный руководитель

д. ф.-м. н., проф.

Белим Сергей Викторович

Омск – 2010

Содержание

Введение

Актуальность работы Надежность современных накопителей информации достаточно высока для обеспечения целостности хранимой информации, однако иногда происходят различные программно-аппаратные сбои, приводящие к повреждению данных на носителе .

В некоторых случаях, как, например, при повреждении мультимедийной информации, повторная передача не требуется, т.к. при неточном восстановлении, возможны лишь некоторые артефакты восприятия восстановленной информации.

Существует несколько моделей представления графической информации, но не все они применимы для восстановления пропущенных данных. В различных моделях предлагается использовать символ пропущенного (поврежденного) значения [1].

В последнее вермя было разработано несколько алгоритмов, применяемых для восстановления различных повреждений графической информации. Наиболее известными являются алгоритм Ричардсона-Люси [2], метод максимальной энтропии [3], алгоритм s-CLEAN [4], адаптивный фильтр [5] и фильтр Винера [6]. Данные алгоритмы позволяют в некоторой степени увеличить резкость изображения и исправить мелкие повреждения достаточно качественно, однако при повреждении нескольких пикселей, а так же повреждении блоков пикселей. Все вышеуказанные алгоритмы восстанавливают изображения с ошибкой, заметной в достаточной мере.

Цели диссертационной работы

1. Разработать модель представления графических данных с помощью матриц с пропусками

2. Исследовать метод восстановления пропущенных данных в графиче-

ской информации с помощью сплайнов

3. Исследовать метод восстановления пропущенных данных в графической информации с помощью кластеризации матриц с пропусками

4. Разработать метод восстановления пропущенных данных в графической информации с помощью самоорганизующихся карт Кохонена

Научная новизна

1. Применение метода кластеризации матриц с пропусками к восстановлению графической информации.

2. Предложена оригинальная модель восстановления графической информации с использованием самоорганизующихся карт Кохонена.

3. Разработан и реализован программный комплекс, выполняющий восстановление изображений, согласно предложенным моделям.

Практическая значимость Результаты, изложенные в диссертации, могут быть использованы для восстановления поврежденных мультимедийных данных. Результаты работы представляют интерес для специалистов в области обработки графической информации.

Основные положения, выносимые на защиту.

1. Модель представления графических данных с помощью матриц с пропусками.

2. Метод восстановления пропущенных данных в графическиой информации с помощью сплайнов.

3. Метод восстановления пропущенных данных в графическиой информации с помощью кластеризации матриц с пропусками.

4. Метод восстановления пропущенных данных в графическиой информации с помощью самоорганизующихся карт Кохонена.

Апробация работы Основные результаты диссертации докладывались и обсуждались на следующих конференциях и семинарах: «Решетнёвские чтения» (2009 г., г.Красноярск), «Информационные технологии и автоматизация управления» (2009 г., г.Омск), внутрифакультетские семинары ФКН ОмГУ 2009-2010 гг.

...

Публикации. Материалы диссертации опубликованы в 7 печатных работах, из них 1 статья в журнале из списка, рекомендованных ВАК [**?**]

Личный вклад автора Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в опубликованные работы. Подготовка к публикации полученных результатов проводилась диссертантом. Все представленные в диссертации результаты получены лично автором.

Структура и объем диссертации Диссертация состоит из введения, 4 глав, заключения и библиографии. Общий объем диссертации 96 страни, включая 14 рисунков и 10 таблиц. Библиография включает 97.

Глава 1

Методы восстановления данных

1.1. Кластеризация данных с пропусками

1.2. Интерполяция сплайнами

Допустим, что необходимо интерполировать значение функции $f(x, y)$ в точке $P(x, y)$, лежащей внутри квадрата $[0, 1] \times [0, 1]$, и известно значение функции f в шестнадцати соседних точках $(i, j), i = -1 \dots 2, j = -1 \dots 2$. Тогда общий вид функции, задающей интерполированную поверхность, может быть записан следующим образом:

$$p(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j. \tag{1.1}$$

Для нахождения коэффициентов a_{ij} необходимо подставить в вышеприведенное уравнение значения функции в известных шестнадцати точках. Например:

$$\begin{aligned} f(-1,0) &= a_{00} - a_{10} + a_{20} - a_{30} \\ f(0,0) &= a_{00} \\ f(1,0) &= a_{00} + a_{10} + a_{20} + a_{30} \\ f(2,0) &= a_{00} + 2a_{10} + 4a_{20} + 8a_{30}. \end{aligned}$$

Полностью в матричном виде:

$$M\alpha^T = \gamma^T,$$

где

$$\alpha = \left[a_{00}\ a_{01}\ a_{02}\ a_{03}\ a_{10}\ a_{11}\ a_{12}\ a_{13}\ a_{20}\ a_{21}\ a_{22}\ a_{23}\ a_{30}\ a_{31}\ a_{32}\ a_{33} \right],$$

$$\begin{matrix}\gamma=[f(-1,-1) & f(0,-1) & f(1,-1) & f(2,-1)\\ f(-1,0) & f(0,0) & f(1,0) & f(2,0)\\ f(-1,1) & f(0,1) & f(1,1) & f(2,1)\\ f(-1,2) & f(0,2) & f(1,2) & f(2,2)],\end{matrix}$$

$$M = \begin{bmatrix} 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 2 & -2 & 2 & -2 & 4 & -4 & 4 & -4 & 8 & -8 & 8 & -8 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 8 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 4 & 4 & 4 & 4 & 8 & 8 & 8 & 8 \\ 1 & 2 & 4 & 8 & -1 & -2 & -4 & -8 & 1 & 2 & 4 & 8 & -1 & -2 & -4 & -8 \\ 1 & 2 & 4 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 4 & 8 & 1 & 2 & 4 & 8 & 1 & 2 & 4 & 8 & 1 & 2 & 4 & 8 \\ 1 & 2 & 4 & 8 & 2 & 4 & 8 & 16 & 4 & 8 & 16 & 32 & 8 & 16 & 32 & 64 \end{bmatrix}. \tag{1.2}$$

Решая получившуюся систему линейных алгебраических уравнений, можно найти значения a_{ij} в явном виде:

$$\alpha^T = \frac{1}{36} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 36 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -12 & 0 & 0 & 0 & -18 & 0 & 0 & 0 & 36 & 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 18 & 0 & 0 & 0 & -36 & 0 & 0 & 0 & 18 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -6 & 0 & 0 & 0 & 18 & 0 & 0 & 0 & -18 & 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & -12 & -18 & 36 & -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & -12 & 2 & 6 & 9 & -18 & 3 & -12 & -18 & 36 & -6 & 2 & 3 & -6 & 1 \\ -6 & -9 & 18 & -3 & 12 & 18 & -36 & 6 & -6 & -9 & 18 & -3 & 0 & 0 & 0 & 0 \\ 2 & 3 & -6 & 1 & -6 & -9 & 18 & -3 & 6 & 9 & -18 & 3 & -2 & -3 & 6 & -1 \\ 0 & 0 & 0 & 0 & 18 & -36 & 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -6 & 12 & -6 & 0 & -9 & 18 & -9 & 0 & 18 & -36 & 18 & 0 & -3 & 6 & -3 & 0 \\ 9 & -18 & 9 & 0 & -18 & 36 & -18 & 0 & 9 & -18 & 9 & 0 & 0 & 0 & 0 & 0 \\ -3 & 6 & -3 & 0 & 9 & -18 & 9 & 0 & -9 & 18 & -9 & 0 & 3 & -6 & 3 & 0 \\ 0 & 0 & 0 & 0 & -6 & 18 & -18 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & -6 & 6 & -2 & 3 & -9 & 9 & -3 & -6 & 18 & -18 & 6 & 1 & -3 & 3 & -1 \\ -3 & 9 & -9 & 3 & 6 & -18 & 18 & -6 & -3 & 9 & -9 & 3 & 0 & 0 & 0 & 0 \\ 1 & -3 & 3 & -1 & -3 & 9 & -9 & 3 & 3 & -9 & 9 & -3 & -1 & 3 & -3 & 1 \end{bmatrix} x^T.$$

Единожды найденные коэффициенты a_{ij} теперь могут быть использованы для многократного вычисления интерполированного значения функции в произвольных точках квадрата $[0, 1] \times [0, 1]$.

1.3. Карты Кохонена

1.4. Метрики сравнения изображений

В ходе работы встал вопрос о сравнении изображений и автоматизации этого процесса.

Метрика Минковского (MM)[7] Норма разницы между изображениями может быть посчитана путем взятия Минковского среднего разниц пикселей сначала пространственно, а затем хроматически (т.е. по зонам):

$$\epsilon^{\gamma} = \frac{1}{K}\sum_{k=1}^{K}\left\{\frac{1}{N^2}\sum_{i,j=0}^{N-1}\left|C_k(i,j) - \hat{C}_k(i,j)\right|^{\gamma}\right\}^{1/\gamma},$$

где $\hat{C}_k(i,j)$ - значение цветосоставляющей пикселя восстановленного изображения.

Для значения $\gamma = 2$ мы получаем формулу для вычисления среднеквадрачиеской ошибки, которая описана ниже.

В ходе работы использовалось значение $\gamma = \infty$:

$$\epsilon^{\infty} = \max_{i,j}\sum_{k=1}^{K}\frac{1}{K}\left|C_k(i,j) - \hat{C}_k(i,j)\right| = \max_{i,j}||\mathbf{C}(i,j) - \hat{\mathbf{C}}(i,j)||$$

Среднеквадратическая ошибка (MSE)[8] является частным случаем метрики Минковского:

$$\epsilon = \frac{1}{N^2}\sum_{i,j=0}^{N-1}\left(C(i,j) - \hat{C}(i,j)\right)^2$$

Метрика разницы с соседями (DON)[9], толерантна к сдвигам пикселей и обычно используется в оченке качества сжатия видеоряда.

$$\epsilon = \sqrt{\frac{1}{2(N-w)^2}\sum_{i,j=w/2}^{N-w/2}\left[A^2 + B^2\right]},$$

$$A = \min_{l,m\in w_{i,j}}\left\{d(\mathbf{C}(i,j), \hat{\mathbf{C}}(i,m))\right\}$$

$$B = \min_{l,m\in w_{i,j}}\left\{d(\hat{\mathbf{C}}(i,j), \mathbf{C}(l,m))\right\}.$$

где $d(\bullet,\bullet)$ - некая функция расстояния. В данной работе рассматривается Эвклидово расстояние.

Нетрудно заметить, что данная метрика, при $w = 1$ сводится к среднеквадратической ошибке, описанной выше. В работе используется размер блока поиска w равный 5.

Метрика многомерного расстояния (MDM)[10], которая основывается на том, что большинство изображений хранятся в больших разрешения (2000 пикселей в каждом измерении и более)

$$\epsilon = \sum_{r=1}^{R} \left(\frac{1}{2^r} \frac{1}{2^{2r-3}} \sum_{i,j=1}^{2^{r-1}} \left[(g_{ij}^R - \hat{g}_{ij}^R)^2 + (g_{ij}^B - \hat{g}_{ij}^B)^2 + (g_{ij}^G - \hat{g}_{ij}^G)^2 \right]^{1/2} \right),$$

где, для примера, g_{ij}^R - среднее значение серой составляющей ij-того блока красной составлюящей, а r - относительное разрешение метрики.

Глава 2

Восстановление графической информации с использованием сплайнов

В современных программах просмотра изображений и видеофайлов для сглаживания неровностей при изменении размеров (передискретизации) изображения используются различные алгоритмы интерполяции. Существует большое количество фильтров для передискретизации[11], такие как интерполяция методом ближайшего соседа, билинейная интерполяция и бикубическая интерполяция.

Из приведенных выше, в работе будет рассматриваться только интерполяция бикубическими сплайнами и производные от нее, т.к. только в этом случае полученные поверхности являются гладкими[11].

2.1. Модель представления графической информации

В ходе работы встал вопрос и представлении графической информации в памяти компьютера для проведения восстановительных вычислений.

В общем случае, изображения состоят из точек, расположенных на узлах прямоугольной сетки. Каждой точке соответствует цвет.

Современные форматы хранения изображений пордразумевают наличие некоторой палитры цветов для данного файла. Палитра характеризуется количеством цветов, которые может принимать каждый пиксель и характеризуется количеством бит, отводимых на номер цвета из палитры. Кроме того, большинство форматов хранят цветовые компоненты каждого цвета - красный, зеленый, голубой (т.н. RGB-палитра).

В рассматриваемое модели, изображение представляется в виде 3-х мат-

риц, количество элементов которых (а так же количество строк и столбцов), соответствует количеству пикселей (ширину и высоту в пикселях) изображения.

Пиксель, который считается поврежденным, представляется в каждой из трех компонентных матриц значением -1.

Символом W' будем обозначать количество столбцов, а символом H' – количество строк в матрицах.

Таким образом, матрицы для изображения 3x3 черных пикселей с поврежденным средним пикселем будут выглядеть следующим образом:

$$\mathbb{R} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbb{G} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbb{B} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Для каждой матрицы введем следующие обозначения для получения значения компоненты пикселя:

1. $\mathbb{R}[x, y]$ - значение красной компоненты, находящейся в x-том столбце на y-той строке, например, $\mathbb{R}[1, 1] = -1$,

2. $\mathbb{G}[x, y]$ - значение зеленой компоненты, находящейся в x-том столбце на y-той строке, например, $\mathbb{G}[1, 1] = -1$,

3. $\mathbb{B}[x, y]$ - значение голубой компоненты, находящейся в x-том столбце на y-той строке, например, $\mathbb{B}[1, 1] = -1$.

Такой подход к представлению изображения в памяти позволяет быстро находить поврежденную область и получать необходимые значения пикселей со всего изображения.

2.2. Восстановление бикубическими сплайнами

Пусть в матрицах $\mathbb{R}$, $\mathbb{G}$ и $\mathbb{B}$ из приведенной выше модели есть один поврежденный пиксель с координатами $(\mathbb{X}, \mathbb{Y})$, т.е. $\mathbb{R}[x, y] = -1$, $\mathbb{G}[x, y] = -1$ и $\mathbb{B}[x, y] = -1$.

Построим функцию вида 1.1 для всех трех компонент:

$$p_r(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij}^r x^i y^j \tag{2.1}$$

$$p_g(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij}^g x^i y^j \tag{2.2}$$

$$p_b(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij}^b x^i y^j \tag{2.3}$$

Дальнейшая задача состоит в нахождении коэфицентов a_{ij}^r, a_{ij}^g и a_{ij}^b.

Для построения бикубического сплайна необходимо 16 значений из неповрежденных пикселей. Для ситуации, когда координаты поврежденного пикселя удовлетворяют следующим условиям:

$$2 \leq x \leq W' - 1, 2 \leq y \leq H' - 1, \tag{2.4}$$

можно использовать простую схему:

$$\begin{pmatrix} \bullet & \circ & \bullet & \circ & \bullet \\ \circ & \bullet & \bullet & \bullet & \circ \\ \bullet & \bullet & \times & \bullet & \bullet \\ \circ & \bullet & \bullet & \bullet & \circ \\ \bullet & \circ & \bullet & \circ & \bullet \end{pmatrix}, \tag{2.5}$$

где $\times$ – поврежденный пиксель, $\circ$ – пиксель, который может быть как поврежденным, так и неповрежденным, а $\bullet$ – пиксель, значение которого

берется для вычислений. Заметим, что пиксели, значения которых берутся для вычислений не должны быть поврежденными ни по одной из компонент.

В случаях же, когда условия 2.4 не выполняются, можно использовать любую другую схему, но следует, однако, придерживаться правила, что выбираемые пиксели для вычисления сплайна должны окружать поврежденный пиксель со всех сторон, если это возможно.

После выборки неповрежденных пикселей необходимо вычислить значения элементов матрицы M (1.2). При вычислении матрицы M, следует придерживаться правила, что в первом столбце внесены коэфициенты при свободных членах. Такой подход позволит некоторым образом ускорить решение СЛАУ в дальнейшем.

Для уменьшения количества решаемых СЛАУ, можно построить матрицу следующего вида:

$$\left(\begin{array}{cccc|ccc} m_{0,0} & m_{1,0} & \dots & m_{16,0} & \mathbb{R}[x'_0, y'_0] & \mathbb{G}[x'_0, y'_0] & \mathbb{B}[x'_0, y'_0] \\ m_{0,1} & m_{1,1} & \dots & m_{16,1} & \mathbb{R}[x'_1, y'_1] & \mathbb{G}[x'_1, y'_1] & \mathbb{B}[x'_1, y'_1] \\ \vdots & & & & & & \\ m_{0,16} & m_{1,16} & \dots & m_{16,16} & \mathbb{R}[x'_{15}, y'_{15}] & \mathbb{G}[x'_{15}, y'_{15}] & \mathbb{B}[x'_{15}, y'_{15}] \end{array}\right), \qquad (2.6)$$

где $m_{i,j}$ - значения матрицы M, x'_i и y'_i - координаты выбранных пикселей, значения которых будут использоваться для вычисления коэфичентов сплайна.

После решения методом Гаусса-Жордана матрица 2.6 будет приведена к следующему виду:

$$\left(\begin{array}{ccccc|ccc} 1 & 0 & 0 & \dots & 0 & \alpha_r[0] & \alpha_g[0] & \alpha_b[0] \\ 0 & 1 & 0 & \dots & 0 & \alpha_r[1] & \alpha_g[1] & \alpha_b[1] \\ 0 & 0 & 1 & \dots & 0 & \alpha_r[2] & \alpha_g[2] & \alpha_b[2] \\ \vdots & & & & & & & \\ 0 & 0 & 0 & \dots & 1 & \alpha_r[15] & \alpha_g[15] & \alpha_b[15] \end{array}\right) \tag{2.7}$$

Подставляя значения $\alpha_r[i]$, $\alpha_g[i]$ и $\alpha_b[i]$ в функции 2.1, 2.2, 2.3 соответственно, мы получим интерполирующие сплайны для всех трех цветовых компонент. Соответственно, значения этих функций в точке $(\mathbb{X}, \mathbb{Y})$ будет соответствовать значению цветовой составляющей пикселя.

В случае, когда поврежден блок пикселей – выбирать пиксели для вычисления сплайна необходимо в окресностях поврежденного блока.

В контексте рассматриваемой задачи, применение данного метода возможно для одиночных поврежденных пикселей. Для поврежденных областей применение бикубической интерполяции приводит к ярковыраженной и заметной невооруженным глазом ошибке восстановления, как показано на рисунке 2.1.

2.3. Восстановление двумерными n-сплайнами

Пусть в матрицах ($\mathbb{R}$, $\mathbb{G}$, $\mathbb{B}$), есть единичный поврежденный участок размерами W на H пикселей, левый верхний угол которого имеет координаты

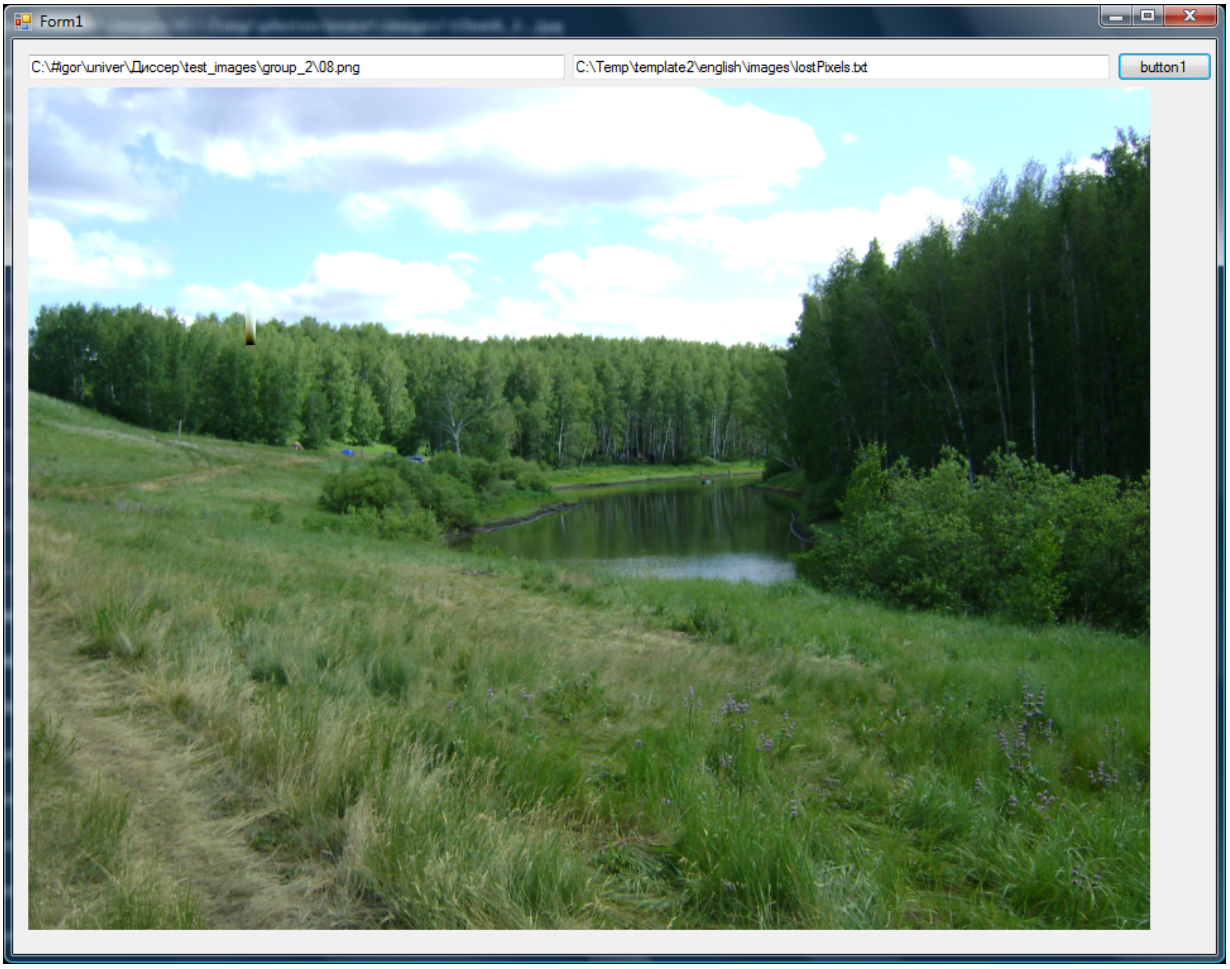

Рис. 2.1. Ошибка восстановления

$(L,\, T)$:

$$\begin{pmatrix}
m_{0,0} & & & \dots & & m_{W',0} \\
\vdots & & & & & \\
m_{0,T-1} & \dots & m_{L-1,T-1} & m_{L,T-1} & \dots & \\
 & \dots & m_{L+W-2,T-1} & m_{L+W-1,T-1} & \dots & m_{W',T-1} \\
 & & & & & \\
m_{0,T} & \dots & m_{L-1,T} & m_{L,T} & \dots & \\
 & \dots & m_{L+W-2,T} & m_{L+W-1,T} & \dots & m_{W',T} \\
 & & & & & \\
\vdots & & & & & \\
m_{0,T+H-2} & \dots & m_{L-1,T+H-2} & m_{L,T+H-2} & \dots & \\
 & \dots & m_{L+W-2,T+H-2} & m_{L+W-1,T+H-2} & \dots & m_{W',T+H-2} \\
 & & & & & \\
m_{0,T+H-1} & \dots & m_{L-1,T+H-1} & m_{L,T+H-1} & \dots &
\end{pmatrix}$$

Опишим алгоритм восстановления блока поврежденных пикселей, основанный на двумерных сплайнах.

В общем случае, функции $p_r(x, y)$, $p_g(x, y)$ и $p_b(x, y)$ будут иметь вид:

$$p_r(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{n} a_{ij}^r x^i y^j \tag{2.8}$$

$$p_g(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{n} a_{ij}^g x^i y^j \tag{2.9}$$

$$p_b(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{n} a_{ij}^b x^i y^j \tag{2.10}$$

Для максимального использования информации, которую несет периметр поврежденного блока, нам необходимо использовать все пиксели, окружающие поврежденный блок (рисунок 2.2).

Темным цветом обозначены поврежденные пиксели, светлым - пиксели периметра поврежденного блока.

Исходя из использования периметра поврежденного блока, сплайн должен иметь следующую размерность:

$$n = \sqrt{2W + 2(H - 1)} - 1.$$

Однако, подкоренное выражение необязательно является квадратом, что приводит к необходимости округления и использования в некоторых случаях пикселей, не входящих в периметр поврежденного блока.

Таким образом, размерность сплайна будет вычислять по формуле:

$$n = \lceil \sqrt{2W + 2(H - 1)} \rceil - 1. \tag{2.11}$$

Количество дополнительных пикселей, необходимых для построения сплайна вычисляются по следующей формуле:

$$k_+ = (n + 1)^2 - (2W + 2(H - 1)),$$

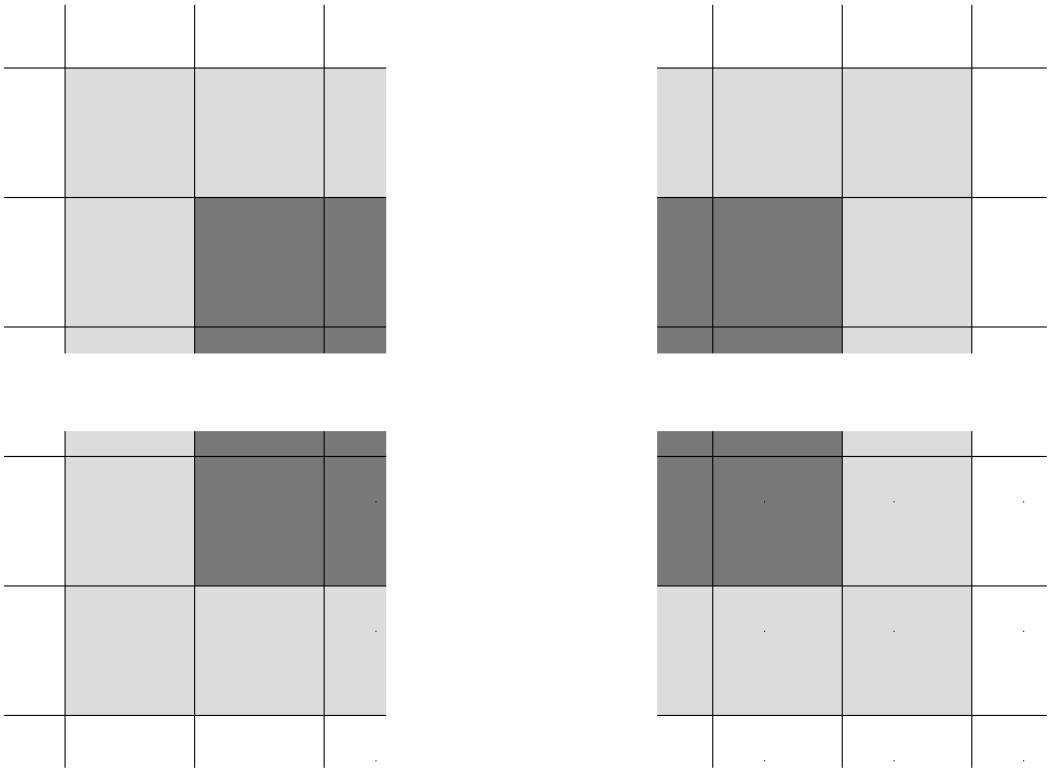

Рис. 2.2. Периметр поврежденного блока

а общее количество используемых пикселей:

$$k = (n+1)^2,$$

Дополнительные пиксели можно выбирать случайным образом из окружения поврежденного блока, однако следует заметить невозможность использования одного пикселя дважды.

Матрица M строится путем вычисления значений при коэфицентах $a_{i,j}$. Строка матрицы M для пикселя с координатами $(x,\, y)$ будет выглядеть следующим образом:

$$\left[\, 1 \;\; x \;\; y \;\; xy \;\; x^2y \;\; xy^2 \;\; x^2y^2 \;\; \ldots \;\; x^{n-1}y^n \;\; x^n y^{n-1} \;\; x^n y^n \,\right].$$

Соответствующие элементы векторов γ_R, γ_G и γ_B будут иметь значения $\mathbb{R}[x, y]$, $\mathbb{G}[x, y]$, $\mathbb{B}[x, y]$ соответственно.

Для уменьшения количества решаемых СЛАУ, можно построить матрицу следующего вида:

$$\left(\begin{array}{cccc|ccc} m_{0,0} & m_{1,0} & \dots & m_{k-1,0} & \mathbb{R}[x'_0, y'_0] & \mathbb{G}[x'_0, y'_0] & \mathbb{B}[x'_0, y'_0] \\ m_{0,1} & m_{1,1} & \dots & m_{k-1,1} & \mathbb{R}[x'_1, y'_1] & \mathbb{G}[x'_1, y'_1] & \mathbb{B}[x'_1, y'_1] \\ \vdots & & & & & & \\ m_{0,k-1} & m_{1,k-1} & \dots & m_{k-1,k-1} & \mathbb{R}[x'_{k-1}, y'_{k-1} & \mathbb{G}[x'_{k-1}, y'_{k-1}] & \mathbb{B}[x'_{k-1}, y'_{k-1}] \end{array}\right), \tag{2.12}$$

где $m_{i,j}$ - значения матрицы M, x'_i и y'_i - координаты выбранных пикселей, значения которых будут использоваться для вычисления коэфицентов сплайна.

После решения методом Гаусса-Жордана матрица 2.6 будет приведена к следующему виду:

$$\left(\begin{array}{ccccc|ccc} 1 & 0 & 0 & \dots & 0 & \alpha_r[0] & \alpha_g[0] & \alpha_b[0] \\ 0 & 1 & 0 & \dots & 0 & \alpha_r[1] & \alpha_g[1] & \alpha_b[1] \\ 0 & 0 & 1 & \dots & 0 & \alpha_r[2] & \alpha_g[2] & \alpha_b[2] \\ \vdots & & & & & & & \\ 0 & 0 & 0 & \dots & 1 & \alpha_r[k-1] & \alpha_g[k-1] & \alpha_b[k-1] \end{array}\right) \tag{2.13}$$

Подставляя значения $\alpha_r[i]$, $\alpha_g[i]$ и $\alpha_b[i]$ в функции 2.8, 2.9, 2.10 соответственно, мы получим интерполирующие сплайны для всех трех цветовых компонент. Соответственно, значения этих функций в точках с кооржинатами, соответствующим координатам поврежденных пикселей, будут соответствовать значениям цветовых составляющих пикселей, находящихся в поврежденном блоке.

Данный алгоритм следует применять к каждой группе поврежденных

пикселей на изображении.

2.4. Результаты восстановления

Для оценки качества восстановления изображений испытания проводились 12 типов.

Изображения можно условно разделить на 4 группы:

1. Высококачественные фотографии NASA,
2. Высококачественные фотографии пейзажей,
3. Фотографии групп людей,
4. Различные искусственные изображения - логотипы, векторная графика, фрактальные изображения, которые были преобразованы в формат BMP.

Было выбрано 20 изображений – по 5 в каждой группе и над каждым ихображением проводилось 100 испытаний с различными степенями повреждений:

1. Поврежден один пиксель,
2. Повреждены 100 одиночных пикселей в различных местах изображения,
3. Повреждена область 8x32 пикселей.

В случае с типом повреждений 1 использовался один бикубический сплайн.

В случае с типом повреждений 2 использовались 100 бикубических сплайнов для каждого поврежденного пикселя.

В случае с типом повреждений 3 использовался один двумерный сплайн степени 9.

Каждое восстановленное изображение сравнивалось с исходным 4-мя метриками, описанными в предыдущей главе.

Результаты испытаний приведены в таблицах 2.1, 2.2 и 2.3 (MM – метрика Минковского, MSE – среднеквадратическая ошибка, DON – метрика разницы с соседями, MDM – метрика многомерного расстояния).

Таблица 2.1. Результаты испытаний. Тип повреждений 1

Метод/Метрика	MM	MSE	DON	MDM
Высококачественные фотографии NASA	3,20	12,14	7,50	56,58
Высококачественные фотографии пейзажей	8,02	16,82	3,59	89,46
Фотографии групп людей	6,48	14,83	6,65	82,46
Искуственные изображения	1,15	8,09	0,92	50,32

Таблица 2.2. Результаты испытаний. Тип повреждений 2

Метод/Метрика	MM	MSE	DON	MDM
Высококачественные фотографии NASA	25,69	65,95	80,64	112,88
Высококачественные фотографии пейзажей	37,29	34,58	30,71	134,76
Фотографии групп людей	31,86	59,23	60,54	145,72
Искуственные изображения	15,82	88,23	22,45	95,92

Таблица 2.3. Результаты испытаний. Тип повреждений 3

Метод/Метрика	MM	MSE	DON	MDM
Высококачественные фотографии NASA	8560,95	36055,48	63345,20	12476,20
Высококачественные фотографии пейзажей	26108,38	2495,62	94347,69	22338,47
Фотографии групп людей	9925,68	43359,23	37438,54	13510,84
Искуственные изображения	10876,29	52848,23	44364,36	12664,67

2.5. Выводы

Метод восстановления изображений путем интерполяции двумерными сплайнами показал себя достаточно быстрым и качественным. Численные результаты испытаний можно оценить если принять во внимание, например, для типа повреждений 2, для высококачественный фотографий пейзажей и метрики Минковского среднее численное значение ошибки составило $37,29$. Если посмотреть на формулу метрики, можно понять, что она показывает количество тонов цвета, на которые отличаются изображения. Приняв во внимание тот факт, что повреждено 100 пикселей, нетрудно посчитать, что в среднем, использование двумерного сплайна размерности 3 на таком типе повреждений приводит к восстановлению с ошибкой в среднем на $0,37$ тона на пиксель. При палитре в 256 оттенков цвета такая ошибка не заметна «на глаз».

Глава 3

Восстановление графической информации методом кластеризации матриц с пропусками

3.1. Восстановление с использованием 2-х и 3-х мерных линейных многообразий

Рассмотрим двемерные и трехмерные линейные многообразия, для которых опишем процедуры построения и ортогонализации, а так же приведем формулы для размерности n.

Однако, хочется заметить, что, как будет показано ниже, использование многомерных линейных многообразий не принесет никакого выигрыша ни в качестве ни в скорости в применении к восстановлению графической информации.

3.1.1. Ортогонализация базисной системы векторов

Пусть дано подпространство размерности n и в нем система линейно независимых векторов $\{y_k\}$, $k = 1..n$. По определению, они образуют базис этого подпространства. Требуется провести процедуру ортогонализации этого базиса, т.е. построить такой новый базис этого подпространства $\{\overline{y}_k\}(k = 1..n)$, чтобы $(\overline{y}_j, \overline{y}_j) = 0$ для любых $i \neq j$.

Для начала решим следующую задачу.

Пусть набор векторов $\{y_k\}(k = 1..n-1)$ уже ортогонализирован, а y_n - еще нет. Тогда построим новый вектор $\overline{y}_n = \sum_{i=1}^{n-q} \alpha_i y_i + y_n$ и потребуем, чтобы $(\overline{y}_n, y_k) = 0$ для всех $k = 1..n-1$. В результате имеем систему из $n-1$

уравнения с $n-1$ неизвестными:

$$\big(\sum_{i=1}^{n}\alpha_i y_i + y_n, y_k\big) = 0, k = 1..n-1.$$

Учитывая, что $(y_i, y_k) = 0$ при $i \neq k$, то легко заметить, что полученная система будет диагональной. Отсюда неизвестные легко находятся по формулам:

$$\alpha_k = -\frac{(y_n, y_k)}{y_k, y_k}.$$

Следовательно, если задана система из n векторов, в которой имеется ортогональная подсистема из $n-1$ вектора, то оставшийся вектор "ортогонализируется"по формуле:

$$\overline{y}_n = y_n - \sum_{i=1}^{n-1}\frac{(y_n, y_i)}{(y_i, y_i)}y_i. \tag{3.1}$$

Тогда процедура ортогонализации системы из n базисных векторов выглядит как последовательная ортогонализация систем из 2, 3, ..., n векторов с использованием 3.1.

3.1.2. Двумерные линейные модели

Для построения двумерного линейного многообразия минимизируем квадратичную форму:

$$\Phi = \sum_{\substack{i,j \\ a_{ij}\neq @}} (a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - b_j) \to \min. \tag{3.2}$$

Решение дается последовательными итерациями по явным формулам. При фиксированных y_{1j}, y_{2j} и b_j значения x_{1i} и x_{2i} однозначно находятся из системы равенств $\partial\Phi/\partial x_{1i} = 0$ и $\partial\Phi/\partial x_{2i} = 0$:

$$\begin{cases} \frac{\partial\Phi}{\partial x_{1i}} = -2\sum\limits_{\substack{j \\ a_{ij}\neq @}} (a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - b_j)y_{1j} = 0, \\ \frac{\partial\Phi}{\partial x_{2i}} = -2\sum\limits_{\substack{j \\ a_{ij}\neq @}} (a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - b_j)y_{2j} = 0. \end{cases}$$

То же самое в векторном виде:

$$\begin{cases} x_{1i}(y_1, y_1)_{a_j} + x_{2i}(y_2, y_1)_{a_i} = (a_i - b, y_1)_{a_i}, \\ x_{1i}(y_1, y_2)_{a_j} + x_{2i}(y_2, y_2)_{a_i} = (a_i - b, y_2)_{a_i}. \end{cases}$$

Аналогично при фиксированных x_{1i} и x_{2i} значения, доставляющие минимум квадратичной форме 3.2, однозначно находятся из равенств $\partial\Phi/\partial y_{1j} = 0$, $\partial\Phi/\partial y_{2j} = 0$ и $\partial\Phi/\partial b_j = 0$:

$$\begin{cases} \frac{\partial\Phi}{\partial y_{1j}} = -2\sum\limits_{\substack{i \\ a_{ij}\neq@}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - b_j)x_{1i} = 0, \\ \frac{\partial\Phi}{\partial y_{2j}} = -2\sum\limits_{\substack{i \\ a_{ij}\neq@}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - b_j)x_{2i} = 0, \\ \frac{\partial\Phi}{\partial b_j} = -2\sum\limits_{\substack{i \\ a_{ij}\neq@}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - b_j) = 0. \end{cases}$$

То же самое в векторной форме:

$$\begin{cases} y_{1j}(x_1, x_1)_{a_j} + y_{2j}(x_2, x_1)_{a_j} + b_j(x_1, 1)_{a_j} = (a_j, x_1)_{a_j}, \\ y_{1j}(x_1, x_2)_{a_j} + y_{2j}(x_2, x_2)_{a_j} + b_j(x_2, 1)_{a_j} = (a_j, x_2)_{a_j}, \\ y_{1j}(x_1, 1)_{a_j} + y_{2j}(x_2, 1)_{a_j} + b_j(1, 1)_{a_j} = (a_j, 1)_{a_j}. \end{cases}$$

Полученная система решается любым численным способом, например, методом квадратного корня (т.к. полученная матрица является симметричной). Учитывая, что полученные векторы y_1 и y_2 могут быть не ортогональны, то в некоторых случаях их необходимо ортогонализировать. А так как они образуют базис подпространства размерности 2, то эта задача легко решается с использованием 3.1.

3.1.3. Трехмерные линейные модели

Для построения трехмерного линейного многообразия минимизируем квадратичную форму:

$$\Phi = \sum_{\substack{i,j \\ a_{ij}\neq@}} (a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)^2 \to \min. \qquad (3.3)$$

Решение дается последовательными итерациями по явным формулам. При фиксированных y_{1j}, y_{2j}, y_{3j} и b_j значения x_{1i}, x_{2i} и x_{3i} однозначно находятся из системы равенств $\partial\Phi/\partial x_{1i} = 0$, $\partial\Phi/\partial x_{2i} = 0$ и $\partial\Phi/\partial x_{3i} = 0$:

$$\begin{cases} \frac{\partial\Phi}{\partial x_{1i}} = -2\sum\limits_{\substack{i \\ a_{ij}\neq @}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)y_{1j} = 0, \\ \frac{\partial\Phi}{\partial x_{2i}} = -2\sum\limits_{\substack{i \\ a_{ij}\neq @}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)y_{2j} = 0, \\ \frac{\partial\Phi}{\partial x_{3i}} = -2\sum\limits_{\substack{i \\ a_{ij}\neq @}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)y_{3j} = 0. \end{cases}$$

То же самое в векторной форме:

$$\begin{cases} x_{1i}(y_1, y_1)_{a_j} + x_{2i}(y_2, y_1)_{a_j} + x_{3i}(y_3, y_1)_{a_j} = (a_j - b, y_1)_{a_j}, \\ x_{1i}(y_1, y_2)_{a_j} + x_{2i}(y_2, y_2)_{a_j} + x_{3i}(y_3, y_2)_{a_j} = (a_j - b, y_2)_{a_j}, \\ x_{1i}(y_1, y_3)_{a_j} + x_{2i}(y_2, y_3)_{a_j} + x_{3i}(y_3, y_3)_{a_j} = (a_j - b, y_3)_{a_j}. \end{cases}$$

Аналогично при фиксированных x_{1i}, x_{2i} и x_{3i} значения, доставляющие минимум квадратичной форме 3.3, однозначно находятся из равенств $\partial\Phi/\partial y_{1j} = 0$, $\partial\Phi/\partial y_{2j} = 0$, $\partial\Phi/\partial y_{3j} = 0$ и $\partial\Phi/\partial b_j = 0$:

$$\begin{cases} \frac{\partial\Phi}{\partial y_{1j}} = -2\sum\limits_{\substack{i \\ a_{ij}\neq @}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)x_{1i} = 0, \\ \frac{\partial\Phi}{\partial y_{2j}} = -2\sum\limits_{\substack{i \\ a_{ij}\neq @}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)x_{2i} = 0, \\ \frac{\partial\Phi}{\partial y_{2j}} = -2\sum\limits_{\substack{i \\ a_{ij}\neq @}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j)x_{2i} = 0, \\ \frac{\partial\Phi}{\partial b_j} = -2\sum\limits_{\substack{i \\ a_{ij}\neq @}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - x_{3i}y_{3j} - b_j) = 0. \end{cases}$$

То же самое в векторной форме:

$$\begin{cases} y_{1j}(x_1, x_1)_{a_j} + y_{2j}(x_2, x_1)_{a_j} + y_{3j}(x_3, x_1)_{a_j} + b_j(x_1, 1)_{a_j} = (a_j - b, x_1)_{a_j}, \\ y_{1j}(x_1, x_2)_{a_j} + y_{2j}(x_2, x_2)_{a_j} + y_{3j}(x_3, x_2)_{a_j} + b_j(x_2, 1)_{a_j} = (a_j - b, x_2)_{a_j}, \\ y_{1j}(x_1, x_3)_{a_j} + y_{2j}(x_2, x_3)_{a_j} + y_{3j}(x_3, x_3)_{a_j} + b_j(x_3, 1)_{a_j} = (a_j - b, x_3)_{a_j}, \\ y_{1j}(x_1, 1)_{a_j} + y_{2j}(x_2, 1)_{a_j} + y_{3j}(x_3, 1)_{a_j} + b_j(1, 1)_{a_j} = (a_j - b, 1)_{a_j}. \end{cases}$$

Полученная система решается любым численным способом, например, методом квадратного корня (т.к. полученная матрица является симметричной).

Учитывая, что полученные векторы y_1, y_2 и y_3 могут быть не ортогональны, то в некоторых случаях их необходимо ортогонализировать. А так как они образуют базис подпространства размерности 3, то эта задача легко решается с использованием 3.1.

3.2. Восстановление с использованием линейных многообразий произвольной размерности

Для построения линейного многообразия произвольной размерности минимизируем квадратичную форму:

$$\Phi = \sum_{\substack{i,j \\ a_{ij} \neq @}} (a_{ij} - \sum_{n} x_{ni} y_{nj} - b_j)^2 \to \min, \tag{3.4}$$

где n - размерность линейного многообразия.

Решение дается последовательными итерациями по явным формулам. При фиксированных y_{nj} и b_j значения x_{ni} однозначно находятся из системы равенств $\partial\Phi/\partial x_{ni} = 0$:

$$\begin{cases} \frac{\partial\Phi}{\partial x_{1i}} = -2 \sum\limits_{\substack{j \\ a_{ij} \neq @}} (a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j) y_{1j} = 0, \\ \frac{\partial\Phi}{\partial x_{2i}} = -2 \sum\limits_{\substack{j \\ a_{ij} \neq @}} (a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j) y_{2j} = 0, \\ \vdots \\ \frac{\partial\Phi}{\partial x_{ni}} = -2 \sum\limits_{\substack{j \\ a_{ij} \neq @}} (a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j) y_{nj} = 0. \end{cases}$$

То же самое в векторном виде:

$$\begin{cases} x_{1i}(y_1, y_1)_{a_j} + x_{2i}(y_2, y_1)_{a_i} + \cdots + x_{ni}(y_n, y_1)_{a_i} = (a_i - b, y_1)_{a_i}, \\ x_{1i}(y_1, y_2)_{a_j} + x_{2i}(y_2, y_2)_{a_i} + \cdots + x_{ni}(y_n, y_2)_{a_i} = (a_i - b, y_2)_{a_i}, \\ \vdots \\ x_{1i}(y_1, y_n)_{a_j} + x_{2i}(y_2, y_n)_{a_i} + \cdots + x_{ni}(y_n, y_n)_{a_i} = (a_i - b, y_n)_{a_i}. \end{cases}$$

Аналогично при фиксированных x_{ni} значения, доставляющие минимум квадратичной форме 3.4, однозначно находятся из равенств $\partial\Phi/\partial y_{nj} = 0$ и $\partial\Phi/\partial b_j = 0$:

$$\begin{cases} \frac{\partial\Phi}{\partial y_{1j}} = -2\sum\limits_{\substack{i \\ a_{ij}\neq@}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j)x_{1i} = 0, \\ \frac{\partial\Phi}{\partial y_{2j}} = -2\sum\limits_{\substack{i \\ a_{ij}\neq@}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j)x_{2i} = 0, \\ \vdots \\ \frac{\partial\Phi}{\partial y_{nj}} = -2\sum\limits_{\substack{i \\ a_{ij}\neq@}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j)x_{ni} = 0, \\ \frac{\partial\Phi}{\partial b_j} = -2\sum\limits_{\substack{i \\ a_{ij}\neq@}}(a_{ij} - x_{1i}y_{1j} - x_{2i}y_{2j} - \cdots - x_{ni}y_{nj} - b_j) = 0. \end{cases}$$

То же самое в векторной форме:

$$\begin{cases} y_{1j}(x_1, x_1)_{a_j} + y_{2j}(x_2, x_1)_{a_j} + \cdots + y_{nj}(x_n, x_1)_{a_j} + b_j(x_1, 1)_{a_j} = (a_j, x_1)_{a_j}, \\ y_{1j}(x_1, x_2)_{a_j} + y_{2j}(x_2, x_2)_{a_j} + \cdots + y_{nj}(x_n, x_2)_{a_j} + b_j(x_2, 1)_{a_j} = (a_j, x_2)_{a_j}, \\ \vdots \\ y_{1j}(x_1, x_n)_{a_j} + y_{2j}(x_2, x_n)_{a_j} + \cdots + y_{nj}(x_n, x_n)_{a_j} + b_j(x_n, 1)_{a_j} = (a_j, x_n)_{a_j}, \\ y_{1j}(x_1, 1)_{a_j} + y_{2j}(x_2, 1)_{a_j} + \cdots + y_{nj}(x_n, 1)_{a_j} + b_j(1, 1)_{a_j} = (a_j, 1)_{a_j}. \end{cases}$$

Полученная система решается любым численным способом, например, методом квадратного корня (т.к. полученная матрица является симметричной). Учитывая, что полученные векторы y_n могут быть не ортогональны, то в некоторых случаях их необходимо ортогонализировать. А так как они образуют базис подпространства размерности n, то эта задача легко решается с использованием 3.1.

3.3. Результаты восстановления

В качестве эталонного метода интерполяции был выбран метод бикубических сплайнов, как наиболее ресурсоёмкий и наиболее распространенный при обработке изображений. В ходе работы было принято правило, что существует некая карта утраченных пикселов. Это было сделано для упрощения

алгоритмов. На практике, определение таковых потерь не является сложным, однако, в контексте рассматриваемой темы, данный вопрос не является критичным.

Были реализованы два метода восстановления потерь и 4 метода сравнения восстановленных изображений с эталонным. Для упрощения вычислений метрики вычислялись только по карте потерь.

Было проведено 100 испытаний на каждом из 20 изображений различных типов:

1. Высококачественные фотографии NASA.
2. Высококачественные фотографии пейзажей.
3. Фотографии групп людей
4. Различные искусственные изображения - логотипы, векторная графика, фрактальные изображения, которые были преобразованы в формат BMP.

Данные типы изображений были выбраны из соображений наиболее полного охвата возможных входных наборов данных.

Для каждого испытания выбирался случайный потерянный блок размером в 64Кб. Причем, начало блока не было кратно ни 64Кб (как номер блока от начала файла), либо 4 байт (как номер потерянного пикселя относительно начала). После выбора границ блока составлялась карта потерянных данных, основываясь на структуре файла и на тех пикселях, любая составляющая которых входила в потерянный блок. Для приближения к реальным условиям, пиксель, в котором хотя бы одна составляющая потеряна признавался потерянным не полностью и обрабатывался отдельно, не прибегая к цветовой модели YUV.

Результаты, пролучекнные в ходе испытаний представлены в таблице 3.1(результат в отдельной группе взят как среднее арифметическое всех испытаний этой группы).

Таблица 3.1. Результаты испытаний

Метрика	MM		MSE	
Метод	Класт.	Спл.	Класт.	Спл.
Высококачественные фотографии NASA	246,51	12,14	6345,20	276,58
Высококачественные фотографии пейзажей	645,02	34,58	9437,69	234,76
Фотографии групп людей	234,76	59,23	4753,29	347,60
Искуственные изображения	96,45	88,23	87,65	62,37
Метрика	DON		MDM	
Метод	Класт.	Спл.	Класт.	Спл.
Высококачественные фотографии NASA	876,45	23,48	758,45	90,23
Высококачественные фотографии пейзажей	589,32	64,75	923,47	65,89
Фотографии групп людей	289,34	75,98	234,76	59,82
Искуственные изображения	85,23	56,56	96,27	98,43

Таблица разбита на 4 группы по 2 столбца в каждой. Каждая группа отображает результаты соответствующей метрики: MM – метрика Минковского; MSE – среднеквадратическая ошибка; DON – метрика разницы с соседями; MDM – метрика многомерного расстояния.

Из таблицы с результатами видно, что кластеризация данных с пропусками по всем статья проигрывает интерполяции бикубическими сплайнами,

кроме случаев искусственных изображений. Это можно объяснить, что в искусственных изображениях доля линейных составляющих велика, в отличие от изображений реального мира.

Результаты испытания методов показали, что кластеризация матриц с пропусками не является достаточно качественным методом для восстановления фотографий. Однако для восстановления изображений, созданных искусственно, данный метод вполне подходит и показывает результаты восстановления не уступающие в достаточной мере методу интерполяции бикубическими сплайнами.

3.4. Оценка эффективности увеличения размерности многообразия

В ходе работы был реализован класс MatrixSolver, который при заданной размерности многообразия строил вектора x_n, y_n и b и производил итерационное исчерпание матрицы данных, как это описано в работе [2].

Итерационные вычисления проводились согласно приведенным формулам многомерных линейных многообразий, приведенных в векторной форме. Данный подход позволил строить СЛАУ, решение которых не вызывало серьезных технических и временных затрат. Все вычисления на каждой итерации сводилось к построению и решению i СЛАУ порядка n и j СЛАУ порядка $n + 1$.

Были проведены испытания для размерностей 2, 3 ... 10. Исходя из полученных на тот момент результатов, были проведены испытания для размерности 100. Все испытания проводились на одним и тем же изображением, имеющим размеры 1024x768 пикселей, что позволяет судить о количестве решенных СЛАУ и их размерности.

Оценка ошибки производилось методом среднего квадратичного [4]. Дан-

ная метрика была выбрана как наиболее наглядная и простая для вычисления. Кроме того, в рамках данной работы, ключевым моментом является выявление зависимости размерности линейного многообразия и таких его свойств, как время, затраченное на достижение определенной, наперед заданной, максимальной ошибки.

Для начала приведем зависимости ошибки восстановления на первых 10-ти шагах восстановления для разных размерностей в таблицах 3.2 и 3.3.

Таблица 3.2. Ошибка восстановления при размерностях $2, \ldots, 6$

i/n	2	3	4	5	6
1	250691595,68	113356199,79	64534470,18	41384009,45	27736091,44
2	125345797,84	56678099,89	32267235,09	20692004,72	13868045,72
3	83563865,23	37785399,93	21511490,06	13794669,82	9245363,81
4	62672898,92	28339049,95	16133617,54	10346002,36	6934022,86
5	50138319,14	22671239,96	12906894,04	8276801,89	5547218,29
6	41781932,61	18892699,96	10755745,03	6897334,91	4622681,91
7	35813085,10	16193742,83	9219210,03	5912001,35	3962298,78
8	31336449,46	14169524,97	8066808,77	5173001,18	3467011,43
9	27854621,74	12595133,31	7170496,69	4598223,27	3081787,94
10	25069159,57	11335619,98	6453447,02	4138400,94	2773609,14

Далее, для наглядности приведем график 3.1 среднеквадратичной ошибки на первых 10-ти итерациях для $n = 2 \ldots 10, 100$:

В ходе испытаний, было выявлено минимальное значение ошибки, которого удалось достичь при восстановлении изображений - 4760. После достижения данного значения при всех рассматриваемых n алгоритм останавливался, т.к. межшаговое падение ошибки было достаточно малым [2].

Приведем длительности одной итерации в таблице 3.4.

Таблица 3.3. Ошибка восстановления при размерностях $7, \ldots, 10, 100$

i/n	7	8	9	10	100
1	20368692,15	15482141,30	10264537,78	3636251,88	51516,38
2	10184346,07	7741070,65	5132268,89	1818125,94	25758,19
3	6789564,05	5160713,77	3421512,59	1212083,96	17172,13
4	5092173,04	3870535,33	2566134,44	909062,97	12879,10
5	4073738,43	3096428,26	2052907,56	727250,38	10303,28
6	3394782,02	2580356,88	1710756,30	606041,98	8586,06
7	2909813,16	2211734,47	1466362,54	519464,55	7359,48
8	2546086,52	1935267,66	1283067,22	454531,48	6439,55
9	2263188,02	1720237,92	1140504,20	404027,99	5724,04
10	2036869,21	1548214,13	1026453,78	363625,19	5151,64

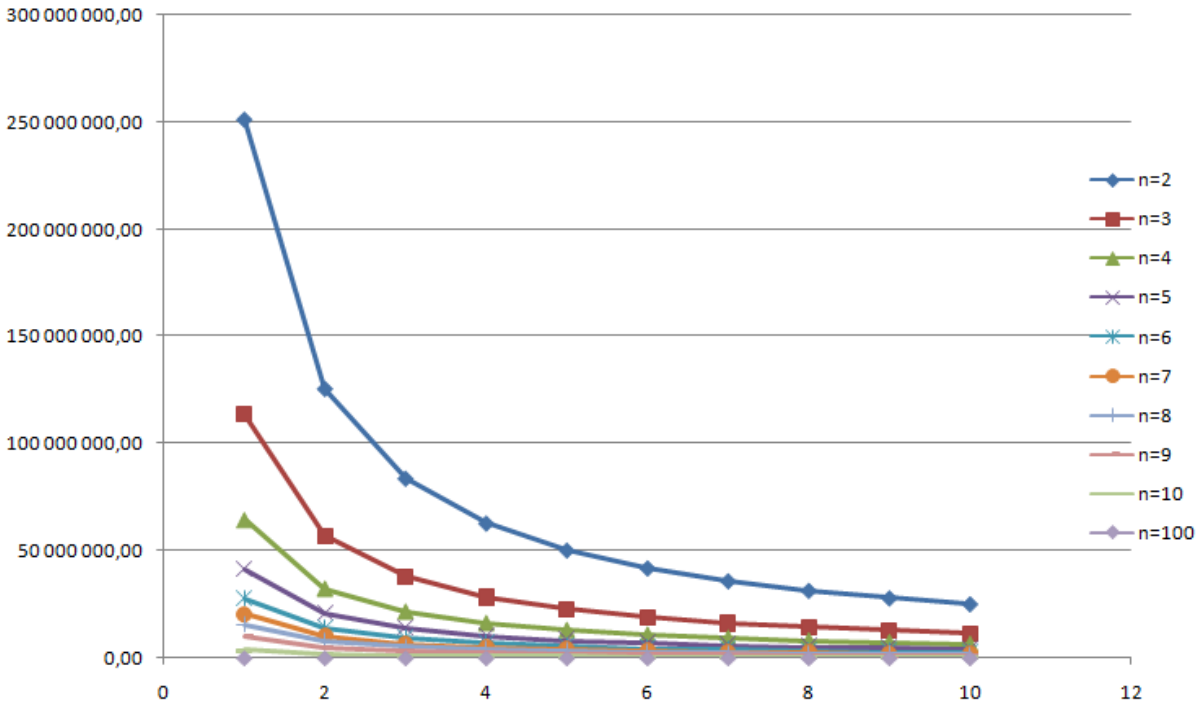

Рис. 3.1. Среднеквадратическая ошибка на первых 10-ти итерациях

Таблица 3.4. Длительность первой итерации

n	Длительность, мс
2	52
3	115
4	202
5	315
6	470
7	640
8	842
9	1270
10	3585
100	153045

Как видно из этой таблицы, длительность одной итерации пропорциональна квадрату размерности многообразия, что говорит о малом количестве накладных расходов на построение матриц и их решение.

Для оценки эффективности применения линейных многообразий больших порядков, рассмотрим изменение ошибки восстановления за единицу времени.

Вычисления производятся по формуле

$$\partial\Phi_i = \frac{\Phi_{i-1} - \Phi_i}{\Delta t},$$

где Φ_i - ошибка восстановления на i-й итерации, а Δt - время выполнения итерации. Сведем данные в таблицу 3.5.

Как видно из таблицы, эффективность падает не только от итерации к итерации, но и с ростом размерности n.

Приведем время, тербуемое для остановки алгоритма согласно критери-

Таблица 3.5. Временная эффективность алгоритма

i/n	2	3	4	5	6
2	2410496,11	492853,04	159738,79	65688,90	29506,48
3	803498,70	164284,35	53246,26	21896,30	9835,49
4	401749,35	82142,17	26623,13	10948,15	4917,75
5	241049,61	49285,30	15973,88	6568,89	2950,65
6	160699,74	32856,87	10649,25	4379,26	1967,10
7	114785,53	23469,19	7606,61	3128,04	1405,07
8	86089,15	17601,89	5704,96	2346,03	1053,80
9	66958,23	13690,36	4437,19	1824,69	819,62
10	53566,58	10952,29	3549,75	1459,75	655,70
i/n	7	8	9	10	100
2	15913,04	9193,67	4041,16	507,15	0,10
3	5304,35	3064,56	1347,05	169,05	0,03
4	2652,17	1532,28	673,53	84,52	0,02
5	1591,30	919,37	404,12	50,71	0,01
6	1060,87	612,91	269,41	33,81	0,01
7	757,76	437,79	192,44	24,15	0,00
8	568,32	328,35	144,33	18,11	0,00
9	442,03	255,38	112,25	14,09	0,00
10	353,62	204,30	89,80	11,27	0,00

ям остановки для каждой размерности в таблице 3.6.

Результаты испытаний наглядно показывают, что данный метод может применяться для восстановления графической информации, однако использование больших размерностей не приводит ни к ускорению восстановления, ни к улучшению качества восстановления.

Таблица 3.6. Время работы алгоритма, MM:CC

n	Время работы, с
2	11:53
3	11:54
4	11:54
5	11:54
6	11:55
7	11:55
8	11:56
9	11:59
10	12:05
100	15:23

Кроме того, увеличение размерности линейного многообразия влечет за собой только увеличение длительности выполнения алгоритма. А так же, уменьшение количества итераций, требуемых для остановки алгоритма. Однако, во всех рассмотренных случаях алгоритм останавливался из-за малого падения ошибки за итерацию и во всех случаях значение конечной ошибки было не менее 5000.

Из всего вышесказанного можно сделать вывод, что восстановление изображений при помощи итерационного исчерпания матрицы с пропусками с использованием линейных многомерных многообразий следует проводить только в 2-хмерном случае.

3.5. Выводы

Глава 4

Восстановление графической информации с использованием карт Кохонена

Карты Кохонена являются очень мощным инструментом для проведения кластеризации различного рода данных. В работе [12] описаны эксперименты по кластеризации звуковой информации. В данной работе описаны эксперименты с графической информацией по восстановлению поврежденных участков. Метод основан на кластеризации данных с имеющихся неповрежденных участков изображения и последующем восстановлении поврежденных участков.

4.1. Модель представления графической информации

Карта Кохонена [12] состоит из набора нейронов, количество которых задается аналитиком. Каждый нейрон, в свою очередь, описывается двумя векторами:

- Вектор веса нейрона m. В общем случае, размерность вектора совпадает с размерностью векторов входных данных.
- Вектор координат p. В общем случае, вектор p описывает точку в пространстве, в котором расположены нейроны.

Обычно нейроны располагают на плоскости в вершинах регулярной решетки с квадратными либо шестиугольными ячейками.

Существуют несколько способов начальной инициализации карты, путем присвоения всем векторам весов нейронов m следующими способами:

- Задание всех координат случайными числами.

- Присваивание вектору веса значения случайного наблюдения из входных данных.

- Выбор векторов веса из линейного пространства, натянутого на главные компоненты набора входных данных.

Следует заметить, что каждый нейрон после инициализации становится неподвижен на карте, т.е. вектор p не изменяется в течение всего обучения.

В общем случае, изображение, представленное в цифровом виде является набором пикселей (точек изображения). При этом, изображение является прямоугольным и характеризуется шириной в пикселях, высотой в пикселях и количеством значений, которое каждый пиксель может принимать (глубина палитры, зачастую измеряется в битах, отведенных на один пиксель). Так же изображение характеризуется методом сжатия (если таковой используется), но в данной работе это не будет рассматриваться.

В силу того, что изображение является прямоугольным, и глаз воспринимает не отдельные точки, а некоторую совокупность, следует рассматривать не последовательный набор пикселей, а некоторую совокупность, сгруппированную вокруг определенной, заранее выбранной точки.

Кроме того, палитра, как набор значений цветов не удовлетворяет модели с поврежденными пикселями, поэтому необходимо ее расширить, путем добавления цвета поврежденного значения.

Для дальнейшего исследования, введем следующие понятия и обозначения:

- Пиксель - номер цвета из палитры изображения, отвечающий соответствующей точке на изображении,

- Поврежденный пиксель - отстствие номера цвета из палитры изображения соответствующего точке на изображении, в силу повреждения

данной точки,

- $\mathbb{G}$ - палитра исходного изображения. В общем случае, $\mathbb{G} \subset \mathbb{Z}$,
- $\sharp$ - цвет, соответствующий поврежденному пикселю,
- $\mathbb{B} = \mathbb{G} \cup \{\sharp\}$ - расширенная палитра, содержащая в себе цвет поврежденного пикселя,
- $x[i]$ - i-я компонента вектора x,
- $W_p \in \mathbb{Z}$ - количество столбцов исходного изображения (ширина изображения в пикселях),
- $H_p \in \mathbb{Z}$ - количество строк исходного изображения (высота изображения в пикселях),
- $P[i, j] \in \mathbb{B}$ - пиксель, расположенный в i-том столбце и j-той строке на изображении,
- $R[i, j] \in \mathbb{G}$ - восстановленное значение пикселя расположенного в i-том столбце и j-той строке на изображении,
- $a \div b, a \in \mathbb{Z}, b \in \mathbb{Z}$ - целая часть деления a на b с остатком,
- $a \oplus b, a \in \mathbb{Z}, b \in \mathbb{Z}$ - остаток от деления a на b,
- $\mathbb{Q}_x$ - ширина карты в нейронах,
- $\mathbb{Q}_y$ - высота карты в нейронах,
- $\mathbb{Q}$ - количество нейронов в карте.
- $m\{x\} \in (0 \ldots \mathbb{Q}_x - 1)$ - горизонтальная координата местоположения нейрона m,

- $m\{y\} \in (0 \dots \mathbb{Q}_y - 1)$ - вертикальная координата местоположения нейрона m.

В ходе разработки модели был выбран следующий способ построения вектора значений из блока пикселей:

$$x[i](x, y) = P[l + i \oplus S, t + i \div S],$$

$$l = x - \lfloor S/2 \rfloor, t = y - \lfloor S/2 \rfloor,$$

где $S \in \mathbb{Z}$ - размер блока, принятый в модели,
$x, y \in \mathbb{Z}$ - координаты пикселя, для которого строится вектор наблюдения.

Соответственно при использования описанного метода построения векторов, необходимо учитывать следующие ограничения:

1. x должно удовлетворять требованию $\lfloor S/2 \rfloor \leq x \leq W_p - \lfloor S/2 \rfloor$,
2. y должно удовлетворять требованию $\lfloor S/2 \rfloor \leq y \leq H_p - \lfloor S/2 \rfloor$,
3. S должно быть нечетным.

Первые два условия вводятся в силу того, что размер рассматриваемого блока больше одного пикселя и, соответственно, количество таких блоков будет меньше, чем количество пикселей в изображении.

Второе условие накладывается для более точного описания окружности пикселя, а именно для симметричности окружности пикселя.

Исходя из вышенаписсаного, можно выделить следующие элементы, для моделирования системы восстановления мультимедийных данных:

1. x - вектор наблюдаемых данных. В применении к восстановлению мультемидийной информации это может быть какой-то набор пикселей (сэмплов, набор блоков сжатия изображений), сгруппированных по определенному правилу.

2. m - вектор веса нейрона. По определению должен быть одной размерности с x.

3. $h_{ci}(t)$ - мера соседства нейронов. Некоторая функция, возвращающая расстояние между нейронами и зависищая от номера итерации t. При определенных условиях может определять и топологию поверхности, на которую нанесена карта.

4. $d(x(t), m_i(t))$ - мера отклонения, показывающая, насколько вектор $x(t)$ не «похож» на вектор $m_i(t)$.

Таким образом, если в некоторой модели определены вышеуказанные элементы, то в этой модели можно использовать самоорганизующиеся карты Кохонена для восстановления поврежденных участков данных.

4.2. Восстановление графической информации с использованием карт Кохонена

Алгоритм обучения производится по итерациям. Пусть t - номер итерации. Положим, что инициализация имела номер итерации 0. Далее выполняются следующие операции:

1. Вибираем случайный вектор $x(t)$ из набора входных значений.

2. Находим расстояния до всех векторов весов всех нейронов карты. Для данной операции выбирается какая-нибудь мера, в общем случае, среднеквадратическое отклонение. Ищем нейрон, наиболее близкий ко входному значению $x(t)$:

$$d(x(t), m_c(t)) \leq d(x(t), m_i(t)),$$

где $m_c(t)$ - вектор веса нейрона победителя (MBU, Winner) $M_c(t)$, $m_i(t)$ - вектор веса i-го узла в карте, $d(x(t), m_i(t))$ - некая мера отклонения. В случае, если вышеуказаному условию удовлетворяет несколько нейронов, BMU выбирается случайным образом.

3. Определение меры соседства нейронов и изменение весов нейронов в карте

   а. Выбираем меру соседства - функцию $h_{ci}(t)$:

   Эта функция определяет «меру соседства» между нейронами M_i и M_c. Обычно, в качестве функции $h_{ci}(t)$ используется гауссовская функция:

$$h_{ci}(t) = \alpha(t) \cdot exp\left(-\frac{||r_c - r_i||^2}{2\delta^2(t)}\right)$$

   де $0 < \alpha(t) < 1$ — обучающий сомножитель, монотонно убывающий с каждой последующей итерацией (то есть определяющий приближение значения векторов веса BMU и его соседей к наблюдению; чем больше шаг, тем меньше уточнение); r_i, r_c — координаты узлов $M_i(t)$ и $M_c(t)$ на карте; $\delta(t)$ — сомножитель, уменьшающий количество соседей с итерациями, монотонно убывает. Параметры α, δ и их характер убывания задаются аналитиком. Более простой способ задания функции соседства:$h_{ci}(t) = \alpha(t)$, если $M_i(t)$ находится в окрестности $M_c(t)$ заранее заданного аналитиком радиуса, и 0 в противном случае. Функция $h(t)$ равна $\alpha(t)$ для BMU и уменьшается с удалением от BMU.

   б. Вычисление ошибки карты:

   Изменяем вектора весов по формуле:

$$m_i(t) = m_i(t-1) + h_{ci}(t)(x(t) - m_i(t-1))$$

Таким образом, все узлы, являющиеся соседями BMU, приближаются к рассматриваемому наблюдению.

4. Выбор остановочного условия.

   Для определения критерия остановки в большинстве случаев используется обишка карты, например, как среднее арифметическое расстояние между наблюдениями и векторами веса соответствующих им BMU:

$$\frac{1}{N}\sum_{i=1}^{N}||x_i - m_c||,$$

   где N - количество элементов набора входных данных.

Мера соседства нейронов $h_{ci}(t)$ представляет собой функцию от индексов нейронов и номера итерации. Для простоты изложения и примерения в дальнейшем представим ее в следующем виде:

$$h_{ci}(t) = h(c, i, t),$$

где h - некоторая, заданная исследователем функция, c - индекс первого нейрона, i - индекс второго нейрона, t - номер итерации.

В ходе исследования, было принято решение испотльзовать видоизмененную функцию Гаусса:

$$h(c, i, t) = \alpha(t) \cdot exp\left(-\frac{d(c, i)^2}{2\delta(t)}\right), \tag{4.1}$$

где $\alpha(t)$ имеет вид:

$$\alpha(t) = \begin{cases} 1, \text{ если t<10,} \\ (t-9)^{-0.2}, \text{ если } t \geq 10, \end{cases} \tag{4.2}$$

$d(c, i)$ имеет вид

$$d(c, i) = \min_{\substack{d_x \in \{-\mathbb{Q}_x, 0, \mathbb{Q}_x\} \\ d_y \in \{-\mathbb{Q}_y, 0, \mathbb{Q}_y\}}} \sqrt{(m_c\{x\} - m_i\{x\} + d_x)^2 + (m_c\{y\} - m_i\{y\} + d_y)^2}, \tag{4.3}$$

$\delta(t)$ имеет вид

$$\delta(t) = 5\frac{\sqrt{\mathbb{Q}}}{\sqrt{t}}. \tag{4.4}$$

Функция $\alpha(t)$ (4.2)на первых 9 итерациях имеет значение 1. Это сделано для того, что бы максимально равномерно инициализировать карту в начале обучения.

Функция (4.3) в приведенном виде будет возвращать расстояние между нейронами так, как будто карта натянута на тороидальную поверхность.

Функция (4.4) приведена к такому виду в ходе экспериментов по обучению карт.

Исследование качества восстановления используя самоорганизующиеся карты Кохоне показало, что данный метод достаточно эффективен и показывает хорошие результаты восстановления. Кроме того0 было выявлено накопление ошибки при восстановлении картами Кохонена от границ поврежденной области к центру.

4.3. Практическая реализация Карт Кохонена для восстановления изображений

В ходе работы был раелизован программный комплекс KohonenMapRestore. В данный комплекс входят:

1. Модуль загрузки и приведения изображений различных форматов ко внутреннему формату программного комплекса,
2. Интерфейс с возможностью выделения поврежденных областей изображения,
3. Модуль нейросети, реализующий функциональность Карт Кохонена,
4. Модуль сохранения и харгузки состояния карты в файл,

5. Модуль оценки качества восстановления, в котором реализованы следующие метрики:

    - метрика Минковского,
    - среднеквадратическая ошибка,
    - метрика разницы с соседями,
    - метрика многомерного расстояния.

Особенности реализации

Так как целью работы было выявление возможности восстановления графической информации с использованием самоорганизующихся карт Кохонен и оценка качества такого восстановления, все объекты в программном комплексе проектировались максимально универсально.

Интерфейс Интерфейс программы (рисунок 4.1) состоит из следующих элементов:

1. Графическая область, в которой можно показать как изображение, так и саму карту. Так же при отображении изображения мажно отмечать поврежденные участки изображения.

2. Текстовое поле с местоположением файла с изображением.

3. Кнопка «loadImage», при нажатии на которую в память приложения загружается указаное в поле ввода изображения и преобразуется во внутренний формат программы.

4. Кнопка «Show Image», при нажатии на которую, в графической области отображается загруженное в данный момент изображение.

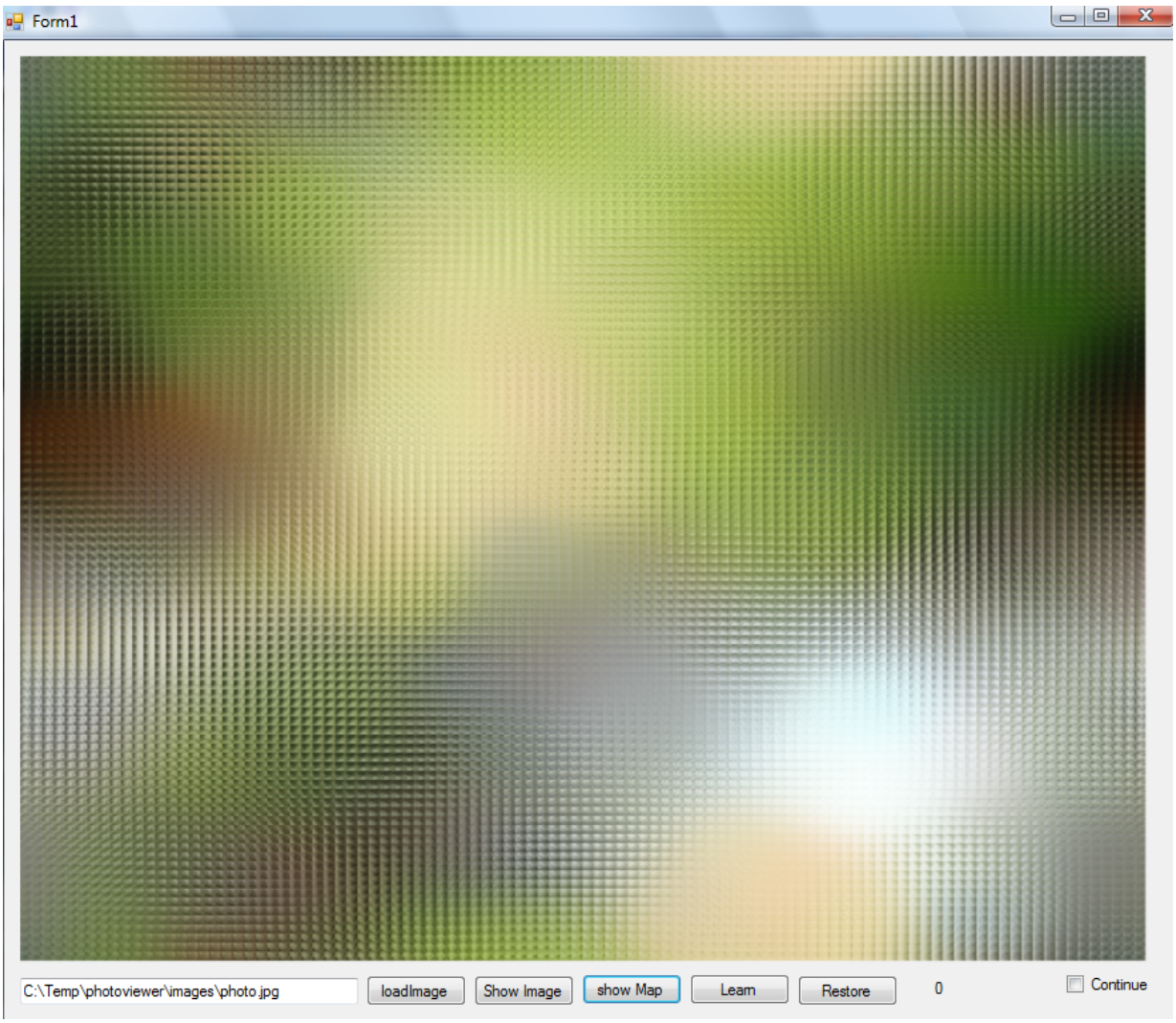

Рис. 4.1. Интерфейс

5. Кнопка «show Map», при нажатии на которую в графической области отображается текущее состояние карты.

6. Кнопка «Learn», нажатие на которую инициализирует процесс обучения.

7. Кнопка «Restore», нажатие на которую инициализирует процесс восстановления изображения.

8. Чекбокс «Continue», который отмечен в процессе обучения. Если в процессе обучения снять галочку, то алгоритм остановися и можно будет

производить восстановление изображения на не до конца обученой карте.

9. Контекстное меню при отображении карты позволяет сохранять и загружать состояние карты из файла.

Нейрон По определению, нейрон обладает следующими характеристиками:

- вектор веса,
- вектор местоположения на карте.

Для максимальной универсальности, вектор веса был реализован как абстрактный объект, а вектор местоположения - как координаты на плоскости. Такая реализация позволяет использовать векторы весов любой размерности.

Мера соседства В силу того, что было принято решение расмещать карту на координатной сетке размерности 2, мера соседства была выбрана как Евклидово расстояние на плоскости.

Однако, в ходе экспериментов было замечено, что при попадании BMU в угол координатной сетки (например:$(0, 0)$, $(\mathbb{Q}_x, \mathbb{Q}_y)$), количество нейронов, которые подвергаются изменению при обучении уменьшается в 4 раза по сравнению с ситуацие, когда BMU попадает в центр карты $(([\mathbb{Q}_x/2], [\mathbb{Q}_y/2]))$.

Из-за этого было принято решение о введении меры соседства, которая бы нивелировала такую проблему, а именно, использование меры соседства вида 4.3. Такое решение позволило избежать проблемы попадания на край карты и максимизировать воздействие обучения.

Функция 4.4 приведена к такому виду в хоте проведения экспериментов на различных размерах карты. Такой вид функции позволил максимизи-

ровать воздействие на нейроны на начальных итерация вне зависимости от количества нейронов.

Функция 4.2 приведена к такому виду для простоты описания процесса инициализации. Первые 10 итераций являются инициализацией и в ходе этих шагов не происходит поиск BMU. Номера нейронов на этом этапе задаюся явно и расположены максимально равномерно по всей карте.

Таким образом, все приведеные выше меры позволили максимально упростить изменение размеров карты при проведении экспериментов и создать наиболее универсальную процедуру инициализации карты.

Процесс обучения Изначально все веса нейронов выставляются в нулевые значения (что соответствует черному цвету в палитре $\mathbb{B}$).

Первые 10 итераций, как и было описано ранее, являются итерациями инициализации. Из изображения выбираются 10 блоков. Несущие максимальное количество информации, т.е. удоавлетворяющие условию:

$$\sum_{\substack{i=x-S/2\ldots-1,1\ldots S/2+x \\ j=y-S/2\ldots-1,1\ldots S/2+y}} (P[i,j]-P[x,y])^2 \to \max,$$

$$P[x,y] \neq \sharp,$$

$$P[i,j] \neq \sharp.$$

Для этих блоков производится обучение карты. При этом местоположение BMU хестко заданы, а именно - равномерно распределены по карте. Это делается для того, что бы изначально карта несла максимальное количество информации из изображения.

На каждой последующей итерации случасйным образом выбирается пара координат (x, y) таким образом, что бы выполнялись следующие условия:

$$\lfloor S/2 \rfloor \leq x \leq W_p - \lfloor S/2 \rfloor,$$

$$\lfloor S/2 \rfloor \leq y \leq H_p - \lfloor S/2 \rfloor,$$

$$P[i,j] \neq \sharp, \text{ при } i = x - S/2 \ldots S/2 + x, j = y - S/2 \ldots S/2 + y.$$

Т.е. из исходного изображения выбираются случайные блоки размером $S \times S$ пикселей, ни один из которых не является поврежденным.

Критерий остановки алгоритма Для остановки обучения можно ввести несколько критериев:

1. Относительно малое изменение карты при обучении на очередной итерации,

2. Исчерпание всех возможных наборов векторов наблюдаемых данных.

В первом случае за критерий остановки было принято условие, при котором ни один из векторов весов нейронов ни в одной из своих компонент не изменил номер цвета. Т.е. $\Delta P < 1$.

Восстановление изображения После остановки процесса обучения можно приступать к самой процедуре восстановления.

Восстановление производится итерационно, как и обучение. При восстановлении из наборов векторов наблюдаемых данных выбирается случайный вектор, в котором поврежденным является только один пиксель.

Для данного вектора ищется BMU. Однако метрика поиска в этом случае изменяется:

$$d(x(t), m_i(t)) = \sum_{\substack{l \\ x(t)[l] \neq \sharp}} (m_i(t)[l] - x(t)[l])^2.$$

Таким образом из всех векторов весовы вибирается наиболее «похожий», без учета компоненты, соответствцующей поврежденному пикселю.

После того, как BMU найден, поврежденный пиксель на изображении заменятеся на соответствующий из вектора веса нейрона.

Данная процедура повторяется од тех пор, пока на исходном изображении не будет ни одного поврежденного пикселя.

Сохранение и загрузка карты Данная функциональность реализована для возможности восстановления по карте, обученной на множестве изображений. В качестве формата файла была выбрана структура XML, которая позволяет хранить сложные структуры в текстовом формате.

Программная реализация В качестве платформы для реализации программного комплекса KohonenMapRestore была выбрана платформа .Net и язык C#, т.к. этот язык позволяет легко оперировать абстрактными объектами и типами данных.

Кроме того, Такой выбор средств реализации позволил сделать удобным выбор метрик, размера блока окружности пикселя, а так же выбор меры соседства из интерфейса приложения.

4.4. Результаты восстановления

Для оценки качества восстановления было произведено сравнение данного метода с методом восстановления изображений с использованием интерполяции бикубическими сплайнами, т.к. этот метод показал хорошие результаты в работе [3].

Для более точной оценки производилось 100 испытаний для каждого из 20 изображений при 3-х различных степенях повреждений:

1. Поврежден один пиксель,
2. Повреждены 100 одиночных пикселей различных местах изображения,
3. Повреждена область 8×32 пикселей.

Изображения можно условно разделить на 4 группы:

1. Высококачественные фотографии NASA,
2. Высококачественные фотографии пейзажей,
3. Фотографии групп людей,
4. Различные искусственные изображения - логотипы, векторная графика, фрактальные изображения, которые были преобразованы в формат BMP.

Каждое испытание проводилось на новой карте поврежденных пикселей. Данная карта наносилась на изображение для обоих методов восстановления, т.о. оценивалось восстановление каждого конкретного случая двумя способами.

Для вычисления ошибки использовалось среднее квадратичное отклонение.

Размер блока S был принят за 7.

Результаты испытаний приведены в Таблице **??**(результат в отдельной группе взят как среднее арифметическое всех испытаний этой группы).

Как видно из таблицы, метод восстановления с использованием карт Кохонена стабильно показывает лучшие результаты, чем метод с бикубическими сплайнами. Стоит заметить, что при некоторых экспериментах в первой группе повреждений карты Кохонена выдавали нулевую ошибку, т.е. абсолютно точно восстанавливали поврежденный пиксель, в то время, как метод с бикубическими сплайнами ошибался в этом же эксперименте на 3-4 оттенка в палитре.

4.5. Выводы

Таблица 4.1. Результаты испытаний

Повреждение	1		2		3	
Метод	Спл.	Карта	Спл.	Карта	Спл.	Карта
Высококачественные фотографии NASA						
Высококачественные фотографии пейзажей						
Фотографии групп людей						
Искуственные изображения						

Заключение

Литература

1. Россиев А. А. Итерационное моделирование неполных данных с помощью многообразий малой размерности: Кандидатская диссертация / КГТУ. 2000.

2. Richardson W. H. Bayesian-Based Iterative Method of Image Restoration // J. Opt. Soc. Am. 1972. Vol. 62. Pp. 55–59.

3. Jaynes E. T. Information Theory and Statistical Mechanics // Physical Review Series II. 1957. Vol. 108. Pp. 620–630.

4. Freedman A., Bose R., Steinberg B. D. Techniques to improve the CLEAN deconvolution algorithm // Journal of the Franklin Institute. 1995. Vol. 332. Pp. 535–553.

5. Hayes M. H. Statistical digital signal processing and modeling. New York: Wiley, 1996.

6. Wiener N. Extrapolation, Interpolation, and Smoothing of Stationary Time Series. New York: Wiley, 1949.

7. Jain A. K. Fundamentals of Digital Image Processing. New Jersey: Prentice Hall, 1989.

8. DiGesu V., Staravoitov V. V. Distance-based Functions for Image Comparison // Pattern Recognition Letters. 1999. Vol. 20. Pp. 207–213.

9. Starovoitov V. V., Köse C., Sankur B. Generalized Distance Based Matching of Nonbinary Images // IEEE International Conference on Image Processing. 1998. Vol. 15. Pp. 83–95.

10. Juffs P., Beggs E., Deravi F. A Multiresolution Distance Measure for Images // IEEE Signal Processing Letters. 1998. Vol. 5. Pp. 138–140.

11. Л. Рабинер Б. . Теория и применение цифровой обработки сигналов. Москва: Мир, 1978.

12. Kohonen T. Self-organizing Maps - 3 ed. Berlin, Heidelberg, New York, Barcelona, Hong Kong, London, Milan, Paris, Singapore, Tokio: Springer, 2001.