Прошин, А.П. Язык обработки изображений и его приложения [текст] / А.П. Прошин, А.М. Данилов, Е.В. Королев, В.А. Смирнов, А.Н. Бормотов // Proceedings of the 5th International Conference “System Identification and Control Problems”, Moscow, 30 jan. – 02 feb. 2006. – Moscow: Institute of Control Sciences, 2006. – pp. 2195…2209

ЯЗЫК ОБРАБОТКИ ИЗОБРАЖЕНИЙ И ЕГО ПРИЛОЖЕНИЯ

А.П. Прошин
Пензенский государственный университет архитектуры и строительства
Россия, 440028, Пенза, Г. Титова ул., 28

А.М. Данилов
Пензенский государственный университет архитектуры и строительства
Россия, 440028, Пенза, Г. Титова ул., 28
E-mail: fmatem@pguas.ru

Е.В. Королев
Пензенский государственный университет архитектуры и строительства
Россия, 440028, Пенза, Г. Титова ул., 28
E-mail: korolev_ev@rambler.ru

В.А. Смирнов
Пензенский государственный университет архитектуры и строительства
Россия, 440028, Пенза, Г. Титова ул., 28
E-mail: kkatarn@rambler.ru

А.Н. Бормотов
Пензенский государственный университет архитектуры и строительства
Россия, 440028, Пенза, Г. Титова ул., 28

Ключевые слова: обработка изображений, алгоритмический язык
Key words: image processing, programming language

Предлагается специализированный алгоритмический язык, предназначенный для описания задач обработки изображений. Язык содержит средства, позволяющие выполнять полутоновые преобразования, цифровую фильтрацию, распознавание признаков в пространственной области и области обратных длин. Приводится описание переносимой реализации интерпретатора предлагаемого языка. Применение разработанного языка возможно как при поисковых исследованиях, так и при решении прикладных задач.

IMAGE PROCESSING LANGUAGE AND ITS APPLICATIONS / A.P. Proshin (Penza State University of Architecture and Construction, 28 G. Titova, Penza, 440028, Russia, E-mail: proshin@gasa.penza.com.ru), A.M. Danilov (Penza State University of Architecture and Construction, 28 G. Titova, Penza, 440028, Russia, E-mail: danilov@gasa.penza.com.ru), E.V. Korolev (Penza State University of Architecture and Construction, 28 G. Titova, Penza, 440028, Russia, E-mail: postmaster@gasa.penza.com.ru), V.A. Smirnov (Penza State University of Architecture and Construction, 28 G. Titova, Penza, 440028, Russia, E-mail: postmaster@gasa.penza.com.ru), A.N. Bormotov (Penza State University of Architecture and Construction, 28 G. Titova, Penza, 440028, Russia, E-mail: postmaster@gasa.penza.com.ru). The special-purpose programming language for the

description of the tasks of image processing is offered. By the means of using offered language it is possible to perform half-tone transformations, digital filtering, pattern recognition both in spatial and inverse lengths domains. Portable implementation of the interpreter of the offered language is described. The usage of the designed language are possible both in exploratory researches and during solution of the applied problems.

1. Введение

Прогресс вычислительной техники сопровождается развитием средств и методов цифровой обработки изображений. Любой метод цифровой обработки в качестве необходимого этапа включает использование вычислительно - логических алгоритмов для преобразования двумерных дискретных образов – *цветовых каналов изображения*. Основные алгоритмов обработки можно разделить на несколько групп.

1. Полутоновые преобразования (в т.ч. – коррекция гаммы и выравнивание гистограмм).

2. Распознавание признаков в пространственной области (построение и последующий анализ деревьев октантов и двоичных деревьев).

3. Распознавание признаков в области обратных длин (вычисление автокорреляционых функций и распознавание образов).

4. Цифровая фильтрация в пространственной области (вычисление свертки); фильтрация в области обратных длин (использование теоремы о свертке), гомоморфные преобразования.

Адекватный решаемым задачам выбор программного обеспечения (ПО), корректно реализующего достаточное число алгоритмов цифровой обработки, может оказаться затруднительным.

2. Альтернативы разработке авторского программного обеспечения

Общей чертой всех методов цифровой обработки изображений является то, что они оперируют *двумерным массивом хроматических координат*. Именно этот объект находится среди входных данных большинства операций, именно он обычно является результатом выполнения операции.

Известны многочисленные программные среды, позволяющие выполнять матричные вычисления (MATLAB [1], Mathcad [2], SciLab [3], Octave [4] и др.). Многие из этих сред лишены диалогового интерфейса, управляются специализированными языками высокого уровня (фактически представляя собой *интерпретаторы* с этих языков) и поддерживают работу в пакетном режиме. При вызове в числе параметров командной строки может быть указано имя файла, содержащего программу на входном языке; дальнейшие преобразования выполняются без вмешательства пользователя.

Функциональность пакета MATLAB доступна через так называемые *наборы инструментальных средств* (*Toolboxes*). Среда MATLAB поставляется с набором, предназначенным для обработки изображений. Средства этого набора позволяют, в частности, выполнять двумерное преобразование Фурье и вычислять двумерную свертку (в том числе – в области обратных длин). Однако функции

для работы с изображениями в различных цветовых пространствах не реализованы.

В программных средах SciLab (версии 2.7), Octave (версии 2.1) и Mathcad (версии 2000 Professional) специализированные средства обработки изображений практически отсутствуют. Octave и Mathcad позволяют лишь получить растровые данные из файла, содержащего полутоновое изображение. В среде SciLab даже эта операция должна быть выполнена средствами управляющего языка. Среда Mathcad не поддерживает пакетный режим работы (что частично компенсируется средствами программирования и автоматическим вычислением рабочего листа; впрочем, предоставляя визуальные средства создания того "кода", который реализует *функциональность* программы, среда Mathcad нарушает саму концепцию создания ПО).

Средства цифровой обработки содержатся в большинстве программных пакетов обработки растровой графики. В частности, пакет GIMP [5] содержит развитые средства перехода между цветовыми пространствами. Возможен статистический анализ информации каждого из цветовых каналов (однако данный анализ выполняется средствами интерпретируемого языка и поэтому весьма требователен к вычислительным ресурсам). Для каждого из цветовых каналов поддерживается вычисление свертки в пространственной области (для версии 1.2 – фильтры размеров до 5x5 включительно).

GIMP допускает работу в пакетном режиме. Входным языком является надмножество LISP - подобного языка UMB Scheme. Синтаксис и семантика этого языка очень сильно отличаются от синтаксиса и семантики наиболее широко распространенных универсальных алгоритмических языков, что может затруднить работу с пакетом.

К сожалению, GIMP (как и большинство других пакетов для работы с растровой графикой) использует однобайтовые целочисленные значения для внутреннего представления изображений. Поэтому, например, выполнение свертки с ядром, сумма элементов которого превышает единицу, приводит к *потере информации*; во многих случаях это неприемлемо. Преобразования в области обратных длин не реализованы (пакет *не создавался* для поисковых и прикладных научных исследований).

Каждый из методов цифровой обработки может быть реализован в виде отдельной программы на универсальном алгоритмическом языке. Такой подход более соответствует исходной философии UNIX - подобных операционных систем: решение задачи осуществляется при помощи набора программ с ограниченной функциональностью – инструментов (*tools, utilities*), объединяемых функциональностью командной оболочки или специализированного интерпретируемого языка (например, Perl). Так, в состав пакета ImageMagic [6] входит инструмент, позволяющий выполнять полутоновые преобразования, переход между цветовыми пространствами и вычисление свертки (с предопределенными или заданными пользователем ядрами фильтров). Инструмент управляется только параметрами командной строки.

Недостатком подхода, использующего инструменты с фиксированной функциональностью, является необходимость повторной реализации всех операций ввода/вывода и перехода между цветовыми пространствами для каждого из инструментов. По всей видимости, указанный недостаток являлся одним из тех, которые заставили авторов пакета ImageMagic включить в состав последних версий интерпретатор с языка Magick Scripting Language (MSL), предназначен-

ного для решения сравнительно сложных задач обработки в пакетном режиме. Язык MSL является производным от метаязыка XML (eXtensible Markup Language) – стиля разметки зависимых от приложений данных. Целесообразность использования языка *разметки* для представления задач цифровой обработки нам представляется спорной.

Язык MSL в настоящее время находится в стадии разработки и его практическое применение затруднительно.

3. Язык обработки изображений и его реализация

Целью настоящего исследования является создание ПО, предоставляющего реализации основных алгоритмов цифровой обработки изображений и отвечающего требованиям:

- корректность реализации;
- поддержка пакетного режима;
- переносимость между разнородными вычислительными платформами;
- открытость, возможность дальнейшего сопровождения и расширения функциональности.

Для достижения поставленной цели решены следующие задачи:

- сформулирован перечень подлежащих реализации алгоритмов;
- выбран вид представления каждой элементарной операции над объектом фундаментального типа;
- выбран вид представления средств, расширяющих функциональность набора основных элементарных операций;
- определены грамматика, синтаксис и семантика языка обработки изображений (далее – язык IPL) ;
- реализован интерпретатор с языка IPL, отвечающий поставленным выше требованиям.

Фундаментальными типами языка являются вектор, матрица и древовидная двоичная декомпозиция изображения. Программа на языке IPL может содержать:

- глобальную секцию;
- описания объектов;
- описания преобразований расширяющих функциональность элементарных операций;
- секцию целей.

Составить представление о возможностях языка позволяет следующий фрагмент формального описания его грамматики (терминальные символы языка – ключевые слова и знаки пунктуации – выделены полужирным шрифтом).

```
IPL-программа:
      IPL-программа | секция
```

```
секция:
      глобальная_секция | определение_объекта |
      определение_расширенного_преобразования |
      определение_функции | секция_целей

определение_объекта:
      определение_вектора | определение_матрицы |
      определение_древовидной_декомпозиции

определение_вектора:
      vector имя_объекта { изменение_состояния | \
            элементарная_векторная_операция }

определение_матрицы:
      matrix имя_объекта { изменение_состояния | \
            элементарная_матричная_операция }

секция_целей:
      targets { цель }

цель:
      цель | целевой_объект | растровое_изображение

целевой_объект:
      имя_типа имя_объекта

имя_типа:
      vector | matrix | decomposition

растровое_изображение:
      output имя_файла цветовое_пространство \
            канал_1 канал_2 канал_3 [канал_4]

цветовое_пространство:
      RGB | RGBA | HSV | CIE

элементарная_векторная_операция:
      элементарная_векторная_операция | копирование_вектора |
      сохранение_вектора | загрузка_вектора |
      циклический_сдвиг | инверсия_порядка_элементов |
      равномерно_распределенный_шум |
      нормально_распределенный_шум |
      установка_всех_элементов | линейная_интерполяция |
      гистограмма | гистограмма_по_диапазону |
      гистограмма_с_записью_координат_центров |
      гистограмма_по_диапазону_с_записью_координат_центров |
      дискретный_интеграл |
      приведение_к_единичному_диапазону |
      приведение_к_заданному_диапазону |
      вызов_преобразования_вектора |
      вызов_преобразования_пары_векторов |
      печать_строки_на_консоли | печать_счетчика_времени
```

```
элементарная_матричная_операция:
      элементарная_матричная_операция | копирование_матрицы |
      сохранение_матрицы | загрузка_матрицы |
      сохранение_в_формате_с_фиксированной_точкой |
      загрузка_из_файла_в_формате_с_фиксированной_точкой |
      загрузка_цветового_канала |
      копирование_подматрицы |
      установка_элемента | установка_всех_элементов |
      установка_элементов_подматрицы |
      циклический_сдвиг_элементов |
      изменение_порядка_элементов |
      дополнение_заданным_значением |
      дополнение_заданным_значением_до_степени_двух |
      равномерно_распределенный_шум |
      нормально_распределенный_шум |
      линейная_интерполяция_по_строкам |
      линейная_интерполяция_по_столбцам |
      прямое_преобразование_фурье |
      обратное_преобразование_фурье |
      вычисление_оценки_спектра_мощности |
      сложение_матриц |
      сложение_всех_элементов_с_константой |
      умножение_на_скаляр | поэлементное_умножение |
      поэлементное_деление |
      обнуление_строк | обнуление_столбцов |
      обнуление_полосы | выделение_полосы |
      приведение_данного_диапазона_контраста_к_единице |
      приведение_полного_диапазона_контраста_к_единице |
      приведение_полного _диапазона_контраста_к_данному |
      нелинейная_коррекция_гаммы |
      отображение_декомпозиции |
      отображение_матрицы_уровней |
      отображение_матрицы_уровней_оттенками_серого |
      вызов_преобразования_матрицы |
      вызов_преобразования_пары_матриц |
      вызов_декартова_преобразования_фурье_образа |
      вызов_декартова_преобразования_пары_фурье_образов |
      вызов_полярного_преобразования_фурье_образа |
      вызов_полярного_преобразования_пары_фурье_образов |
      печать_строки_на_консоли | печать_счетчика_времени

вызов_преобразования_вектора:
      vector_transform имя_преобразования \
            ( void | список_параметров )

список_параметров:
      , список_параметров | вещественный_параметр

загрузка_цветового_канала:
      input имя_файла_растрового_изображения имя_канала

имя_канала:
red | green | blue | alpha |
      cieX | cieY | cieZ |
      hue | saturation | value
```

```
определение_расширенного_преобразования:
      преобразование_вектора |
      преобразование_пары_векторов |
      преобразование_матрицы |
      преобразование_пары_матриц |
      декартово_преобразование_фурье_образа |
      декартово_преобразование_пары_фурье_образов |
      полярное_преобразование_фурье_образа |
      полярное_преобразование_пары_фурье_образов
```

Каждое из определений объекта фундаментального типа дается в отдельной секции. Например, создание матрицы `SChannel`, содержащей канал цветовой насыщенности растрового изображения `Source.tga`, реализуется фрагментом кода

```
matrix SChannel
{
      input "Source.tga" saturation
}
```

Текущее состояние каждого объекта в любой момент может быть зафиксировано или восстановлено из текстового файла (все значения представляются в формате с плавающей точкой).

Подлежащие созданию объекты и растровые изображения должны быть перечислены в секции целей (грамматика приведена выше). Например, создание полутонового растрового изображения `Source_Saturation.tga`, представляющего оттенками серого канал насыщенности изображения `Source.tga`, реализуется фрагментом

```
targets
{
      output "Source_Saturation.tga" RGB SChannel SChannel SChannel
}
```

Интерпретатор с языка IPL реализован в виде автономной программы (распространяется в исходных текстах; текущая альфа-версия доступна на FTP-сервере В.А. Смирнова [8]). В структуре интерпретатора выделены два уровня – прикладной и сервисный.

На прикладном уровне реализованы синтаксический разбор управляющей IPL - программы и основные алгоритмы цифровой обработки (например, прямое и обратное действительно-значные двумерные преобразования Фурье).

Сервисный уровень содержит реализацию абстрактных типов данных; реализацию объектов, изолирующих системные вызовы целевой платформы; реализацию виртуальной файловой системы (семантика схожа с файловыми вызовами платформы POSIX); реализацию вспомогательных алгоритмов (управление памятью; средства диагностики и отладочной печати, и т.д.).

Как основное ПО, так и сервисный уровень реализованы на языке ANSI C. В качестве компилятора на платформе Win32 использован Microsoft C 12.00.8168 (входящий в состав среды Visual C 6.0). На платформе POSIX был использован GCC (работоспособность проверена с версиями 2.95.3 и 3.2.0).

Основная функциональность сервисного уровня доступна прикладному уровню через методы (функции в терминах языка C), принимающие в качестве одного из параметров дескриптор объекта, инкапсулирующего тот или иной

системный вызов. Язык С является процедурно-ориентированным, поэтому представление объектов оказывается несколько искусственным (объект реализован как структура в терминах языка С; невиртуальный метод – как функция, принимающая адрес этой структуры; виртуальный метод – как указатель на аналогичную функцию, содержащийся в самой структуре, и т.д.). Тем не менее практика свидетельствует, что при подобном подходе вычислительная эффективность программы успешно совмещается с хорошей читаемостью и высокой переносимостью ее кода.

4. Приложения

В качестве примера практического применения разработанного ПО можно привести решение двух задач – *оценки эффективности цифрового фильтра* и *обращения свертки*.

Для количественного сравнения результатов цифровой фильтрации выполняется их *двоичной декомпозиция*. Алгоритм начинает работу с анализа всего изображения. Если разность между максимальным и минимальным элементами текущей области не превышает заданного значения – *порога разложения* – то область становится *листом* двоичного дерева; при невыполнении этого условия исследуемая область разделяется на две подобласти и декомпозиция рекурсивно выполняется для каждой из них.

Исходному дискретному образу дерево двоичной декомпозиции ставит в соответствие *матрицу уровней*, элементы связаны с глубиной соответствующего листа. При заданном пороге разложения зависимость числа листов от уровня определяется пространственным распределением элементов. С числом листов связана *заселенность уровня*

$$P(l) = \frac{1}{\ln 2} \operatorname{arsh} N,$$

где N – число листов на уровне l.

Тестовым является синтетическое изображение, поле яркости которого содержит участки различной однородности, при аддитивном шуме, параметры нормального распределения которого $M[X] = 0$, $\sigma = 0{,}1$.

Подавление верхних пространственных частот (ослабление шума) является целью фильтрации. Ядро принимается в виде

$$f(k,s) = \alpha \exp\left(-\frac{(k - N/2)^2 + (s - M/2)^2}{\beta} \right),\ k = \overline{0, N-1}, \tag{1}$$

$s = \overline{0, M-1}$,

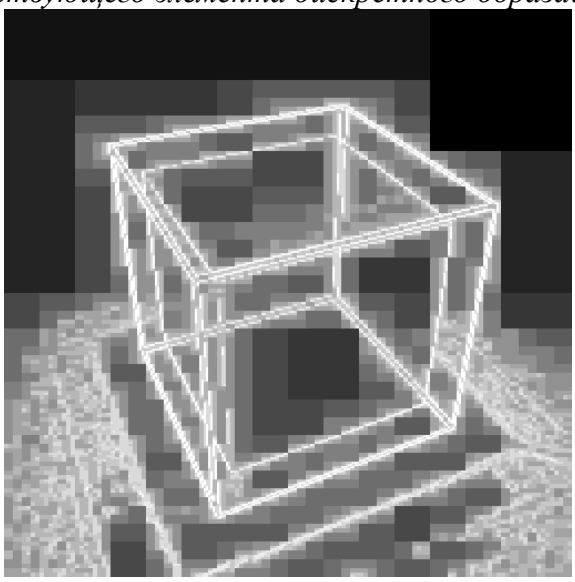
где N, M – размеры фильтра, β – параметр фильтра,

$$\alpha = \left[\sum_{r=0}^{M-1} \sum_{c=0}^{N-1} \exp\left(-\frac{(k - N/2)^2 + (s - M/2)^2}{\beta} \right) \right]^{-1}$$

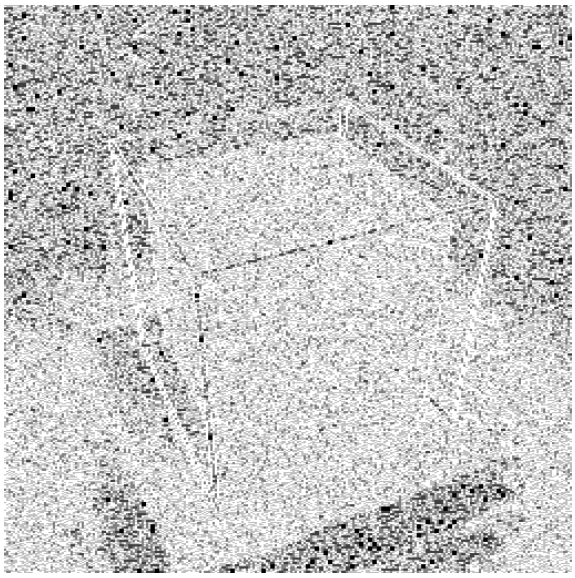
– нормирующий множитель. Сглаживание выполнено фильтрами размеров 3x3 и полноразмерным.

При пороге разложения 0,1 (совпадает со стандартным отклонением шумовой составляющей) осуществлена двоичная декомпозиция исходного, зашум-

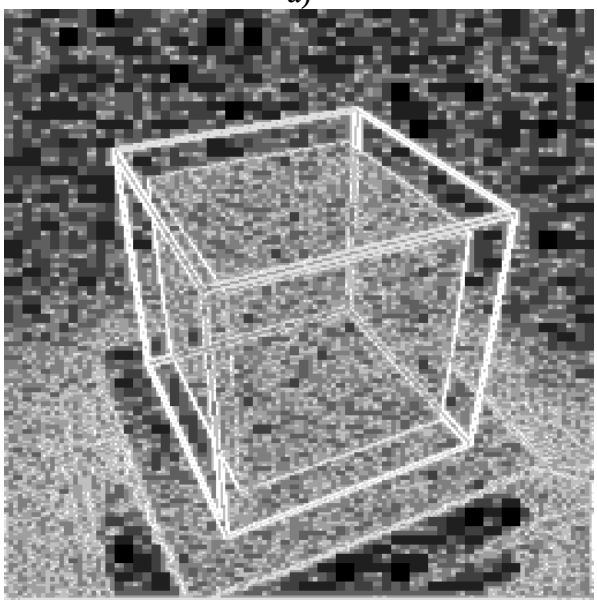
ленного и двух сглаженных изображений. Матрицы уровней приводятся рис. 1; *глубине листа двоичного дерева поставлена в соответствие яркость соответствующего элемента дискретного образа*.



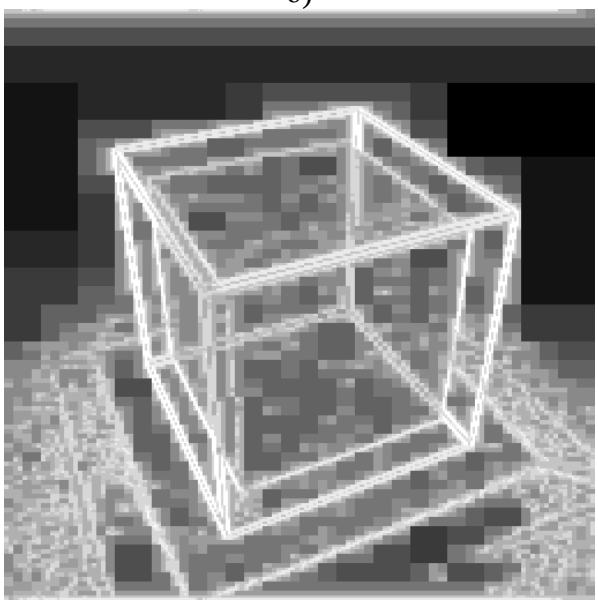
а)



б)



в)



г)

Рис. 1. Матрицы уровней: а) – тестового изображения; б) – зашумленного изображения; в) – сглаживание фильтром 3х3; г) – сглаживание полноразмерным фильтром.

Зависимости заселенности от номера уровня приведены на рис. 2.

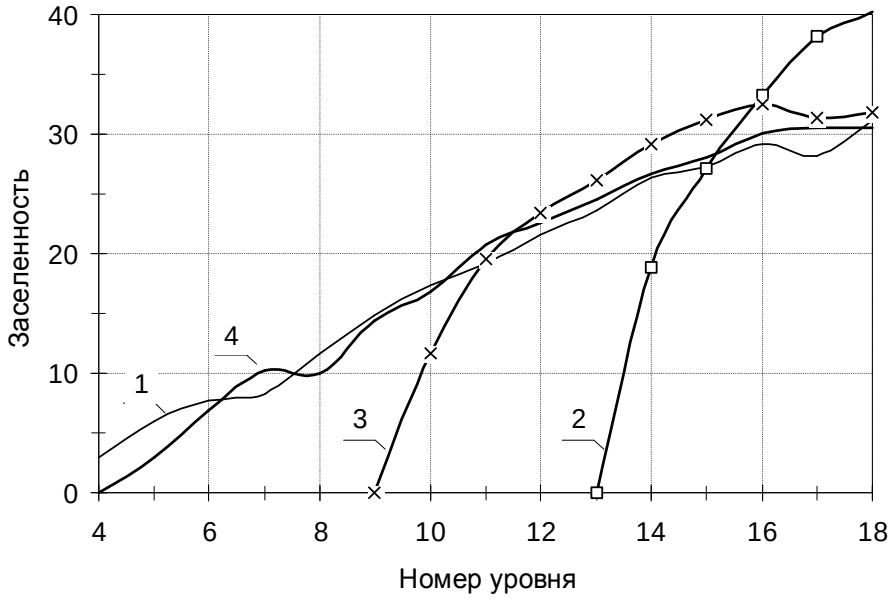


Рис. 2. Зависимость заселенности от номера уровня: 1 – тестовое изображение; 2 – зашумленное изображение; 3 – сглаживание фильтром 3x3; 4 – сглаживание полноразмерным фильтром

Количественной характеристикой различия между двумя изображениями является параметр

$$D = \sqrt{\sum_{l=0}^{l_{\max}} \left(P(l) - P_0(l) \right)^2},$$

где $l_{\max}$ – максимальная глубина дерева, $P_0(l)$ – заселенность уровня l декомпозиции исходного изображения.

Для тестового и зашумленного изображений $D \approx 50$, для тестового и сглаженного фильтром 3x3 – $D \approx 25$, для тестового и сглаженного полноразмерным фильтром – $D \approx 6$. Таким образом, использование полноразмерного фильтра позволяет уменьшить различие между тестовым изображением и результатом сглаживания. Использование фильтра малых размеров не позволяет восстановить детали с наибольшим пространственным масштабом (заселенность на уровнях 4...10 занижена, действие фильтра проявляется только уровнях от 11 до 18, характерный размер элемента на которых близок к размеру фильтра).

Соотношение (1) часто принимается в качестве модели функции рассеяния несовершенного фотоприемного устройства (т.н. *нечеткий детектор*). При регистрации на выходе такого устройства будет получен образ, соответствующий рис. 3.

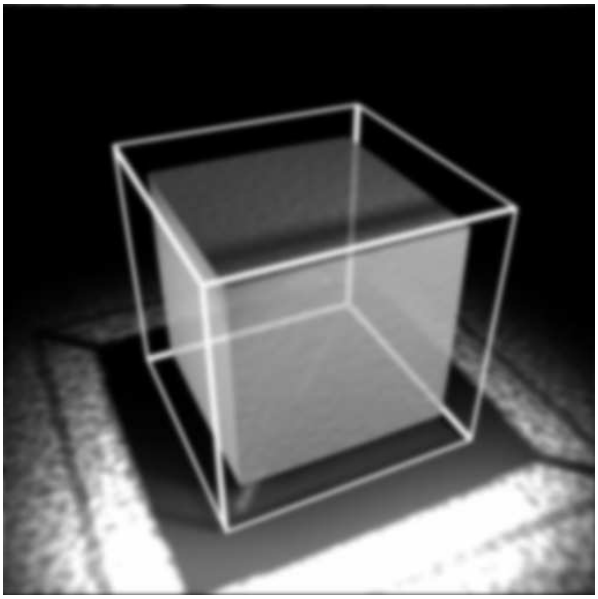


Рис. 3. Результат регистрации нечетким детектором

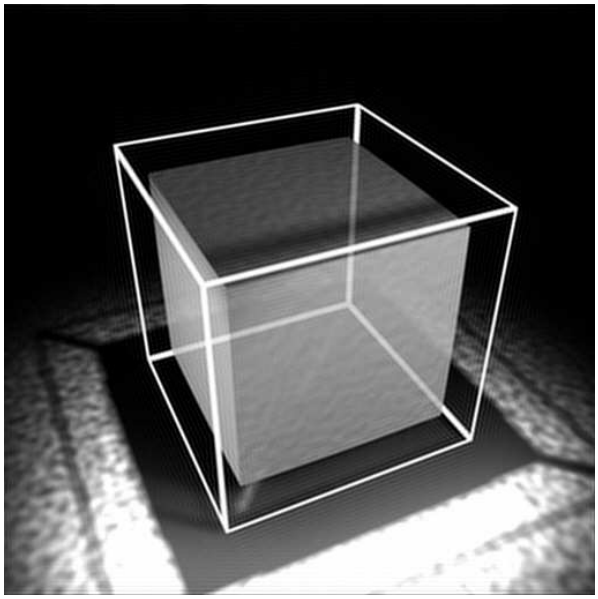


Рис. 4. Восстановленное изображение

Задача восстановления исходного изображения решается обращением свертки. Результат выполнения этой операции показан на рис. 4; отчетливо видны детали (соответствующие реальному изображению), которые *отсутствуют* на оригинале (изображении, полученном на выходе детектора). Ниже приводится код IPL - программы, реализующей численный эксперимент, состоящий в моделировании регистрации изображения нечетким детектором и последующем обращении свертки.

```
global
{
      alias pic "test_cube.jpg"
      alias gkernel_alpha 5
}

matrix src
{
      input pic value
      normalize
}

function half_width
{
      return (width_of src,2,/,floor);
}

function half_height
{
      return (height_of src,2,/,floor);
}

#
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Формирование сглаживающего фильтра с Гауссовым ядром

matrix_transform gaussian_kernel(alpha)
{
      step {
```

```
            var rowc, colc, scale;
            rowc = ($MAX_ROW,1,+,2,/);
            colc = ($MAX_COL,1,+,2,/);
            scale = (-0.5,alpha,/);
      }
      pass {
            var rd, cd;
            rd = ($ROW,rowc,-);
            cd = ($COL,rowc,-);
            $CELL = (rd,rd,*,cd,cd,*,+,scale,*,exp);
      }
      step {
            var sum;
            sum = 0;
      }
      pass {
            sum = (sum,$CELL,+);
      }
      step {
            sum = (1,sum,/);
      }
      pass {
            $CELL = ($CELL,sum,*);
      }
}

#
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~
# Гауссов фильтр

matrix gsmooth_filter
{
      expand width_of src height_of src
      matrix_transform gaussian_kernel(gkernel_alpha)
      shift half_width half_height
      fourier_forward
}

#
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Сглаживание сверткой с Гауссовым фильтром

matrix convolved
{
      copy src
      fourier_forward
      mat_scale gsmooth_filter
      fourier_inverse
}

#
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Обращение свертки

matrix deconvolved
{
      copy convolved
      fourier_forward
      mat_unscale gsmooth_filter
      fourier_inverse
}

targets
{
      output Pict_3.bmp RGB convolved convolved convolved
```

```
    output Pict_4.bmp RGB deconvolved deconvolved deconvolved
}
```

Предполагается, что тестовое изображение содержится в файле `test_cube.jpg`; в матрицу `src` помещается канал яркости этого изображения. Матрица `gsmooth_filter` содержит Фурье-образ сглаживающего фильтра; матрица `convolved` является результатом свертки, а матрица `deconvolved` – результатом ее обращения.

5. Заключение

Предложен язык описания основных операций цифровой обработки изображений и выполнена переносимая реализация интерпретатора с этого языка. При разработке ПО учтены недостатки программных средств аналогичного назначения (пакеты матричных вычислений, пакеты для работы с растровой графикой).

Применение разработанного ПО рекомендуется как в поисковых научных исследованиях, так и при решении прикладных задач. Среди последних – выполнение полутоновых преобразований, распознавание признаков в пространственной области и области обратных длин, цифровая фильтрация и гомоморфные преобразования.

Разработанное ПО распространяется на условиях лицензии GPL.

Список литературы

1. http://www.mathworks.com
2. http://www.mathsoft.com/MATHCAD
3. http://www-rocq.inria.fr/scilab
4. http://www.che.wisc.edu/octave
5. http://www.gimp.org
6. http://www.imagemagick.org
7. Smith S. The Scientist and Engineer's Guide to Digital Signal Processing. – 2nd ed. – San Diego: California Technical Publishing, 1999.
8. Текущий снимок каркасной библиотеки libv [электронный ресурс]. – URL: ftp://sleepgate.ru/devel/libv-current