Абрамов И.А., Банникова А.Н., Мартышкина Н.А. Реализация параллельного обнаружения контуров объектов на изображении. // Проблемы информатики в образовании, управлении, экономике и технике: Сб. статей XVII Междунар. научно-техн. конф. – Пенза: ПДЗ, 2017. – С. 94-98.

УДК 004.42

РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНОГО ОБНАРУЖЕНИЯ КОНТУРОВ ОБЪЕКТОВ НА ИЗОБРАЖЕНИИ

И.А. Абрамов, А.Н. Банникова, Н.А. Мартышкина

MULTI-THREADED IMPLEMENTATION OF LOCAL IMAGE FILTERING

I.A. Abramov, A.N. Bannikova, N.A. Martyshkina

Аннотация. В статье рассматривается один из подходов ускорения выделения контуров объектов на изображении за счёт многопроцессного выполнения, приводятся экспериментальные данные ускорения параллельной программы.

Ключевые слова: обработка изображений, параллельная обработка данных, ускорение алгоритма.

Abstract. The article dealsone of the approaches acceleratethe contour extraction object in the image due to the multi-process execution, the experimental acceleration data of the parallel program.

Keywords: image processing, parallel processing, acceleration of the algorithm.

Обнаружение контуров на цифровых изображениях является одной из задач обработки и анализа изображений. Необходимость в этом возникает в ряде прикладных задач, таких как обработка медицинских и биологических снимков, распознавание объектов в системах технического зрения и многих других задач.

Большая часть алгоритмов обнаружения контурных границ основаны на выделении связной области изображения, яркость пикселей которой различается не более чем на некоторую величину. Контурам изображения будут соответствовать перепады яркости, вызванные разницей уровней яркости фона и объекта. Для обнаружения перепада яркости используется значение градиента $\nabla \mathbf{f}(x, y)$ в точке (x, y), вычисление которого состоит в определении частных производных $\frac{\partial f}{\partial x}$ и $\frac{\partial f}{\partial y}$ точки цифрового изображения [1]. В этом случае предполагается, что изменение яркости точки, расположенной на границе изображения, должно быть значимо больше допустимого изменения яркости в точке фона. Способ определения значимости изменения яркости состоит в задании порога. Тогда точка будет является контурной, если её первая производная превышает заданный порог.

При обнаружении контуров важную роль играет модуль вектора градиента, равный $\nabla f = |\nabla \mathbf{f}| = \sqrt{G_x^2 + G_y^2}$. Этот вектор с осью x составляет уголα(x, y),равный $\alpha(x, y) = \text{arctg}\left(\frac{G_y}{G_x}\right)$. Тогда направление контура в точке (x, y) будет перпендикулярно направлению вектора градиента в этой точке [2].

Один из способов нахождения частных производных $\frac{\partial f}{\partial x}$ и $\frac{\partial f}{\partial y}$ точки цифрового изображения состоит в применении оператора Собела. При этом изображение сканируется окном размером 3×3 (рисунок) и вычисляются значения частных производных по формулам:

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3),$$
и
$$G_x = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7),$$

где z_i – значения окрестности на рисунке.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Окрестность 3×3 внутри изображения (переменные z_i представляют собой значение яркости)

В процессе обнаружения контуров обычно решается задача формирования из множества контурных точек границ объектов. Для этого выполняется анализ характеристик пикселов в окрестности сканирующего окна каждой точки изображения, которая помечена как контурная. Все точки, удовлетворяющие заранее заданным критериям, связываются и образуют контур. Пиксель контура, имеющий координаты (x_0, y_0) и расположенный внутри заданной окрестности точки (x, y), считается сходным по модулю градиента с пикселем (x, y), если

$$|\nabla f(x, y) - \nabla f(x_0, y_0)| \leq E,$$

где E – заданный неотрицательный порог.

Пиксель контура, имеющий координаты (x_0, y_0) и расположенный внутри заданной окрестности точки (x, y), считается сходным по направлению градиента с пикселем (x, y), если

$$|\alpha(x, y) - \alpha(x_0, y_0)| \leq A,$$

где A – заданный неотрицательный угловой порог.

Пиксель в заданной окрестности объединяется с центральным пикселем (x, y), если выполнены критерии сходства по величине и по направлению. Этот процесс повторяется в каждой точке изображения, с одновременным запоминаем найденных связанных пикселей при движении центра окна. Простой способ учета данных состоит в том, что каждому множеству связываемых пикселей контура присваивается своё значение яркости [1].

Время обработки изображений существенно зависит от размеров самого изображения и маски. Можно приблизительно оценить время выполнения алгоритма. Определение значения градиента и его направления в пределах окна размера 3×3 потребует выполнить 15 арифметических операций без учета операций управления циклом. Приняв, что τ_c – это время выполнения одной арифметической операции [3], приблизительное время вычислений внутри заданной окрестности точки равно $15\tau_c$. Тогда, с учётом размера изображения $N \times M$, полное время работы алгоритма на одном процессоре составит

$$T_1 = 15\tau_c N \times M.$$

Алгоритм обнаружения контуров требует выполнения множества однотипных операций над пикселами, что свидетельствует о наличии параллелизма по данным. В этом случае исходное изображение предполагается разбить на горизонтальные полосы и выделить каждому процессу подмножество строк в количестве N/p,

где p – число создаваемых рабочих процессов, а N – размер изображения по вертикали [4].

Для оценки времени выполнения параллельного алгоритма можно предположить, что исходные данные уже распределены по процессорам нужным образом, а по окончании работы результат также распределён по процессорам. Тогда время выполнения алгоритма на системах сp процессоров будет равно $15\tau_c N \times M/p$. Это время учитывает только выполнение вычислительных операций, но не учитывает издержки на получение процессором данных и отправку результата. Пусть на выполнение одной операции передачи данных между процессами требуется время τ_s [3], тогда полное время для получение исходных данных одному процессу и отправки результата составит $2\tau_s N \times M/p$. В этом случае время, затраченное каждым процессом, будет равно

$$T_p = \frac{N \times M(15\tau_c + 2\tau_s)}{p}.$$

Ускорение параллельного алгоритма составит:

$$S_p = \frac{T_p}{T_1} = \frac{N \times M \times 15\tau_c \times p}{N \times M(15\tau_c + 2\tau_s)} = \frac{15p}{15 + 2\frac{\tau_s}{\tau_c}}.$$

Если принять время выполнения процессором арифметической операции примерно равным времени одной операции передачи или получения данных, то ускорение от распараллеливания алгоритма будет равно $S_p = 15p/17$. Для некоторых значений p ожидаемое ускорение алгоритма представлено в таблице 1.

Таблица 1

Приблизительные оценки ускорения алгоритма

Количество вычислительных узлов в системе	**Ускорение**
2	1,76
4	3,53
6	5,29
8	7,05

Для экспериментальной оценки ускорения параллельного алгоритма были разработаны последовательная и параллельная программы. Последовательная программа была реализована на языке C++и скомпилированы с помощью компилятора gcc5.4.0. Параллельная программа также реализована на языке C++ с помощью технологии MPI. Эксперименты с программой проводились на компьютерах на базе процессора IntelCorei3, 3,5 ГГц, 8 Гб ОЗУ под управлением операционной системы Ubuntu 16.04, которые объединены сетью GigabitEthernet. В эксперименте были исследованы последовательная программа, двухпроцессный, четырёхпроцессный, шестипроцессный и восьмипроцессный варианты выполнения параллельной программы.

В процессе эксперимента было обработано с помощью маски 3×3 изображение размером 5000×3000. Результаты вычислительных экспериментов выделения контуров на изображении приведены в таблице 2. Время выполнения указано в миллисекундах.

Таблица 2

Результаты вычислительных экспериментов

Последовательный алгоритм, время, мс	Параллельный алгоритм							
	2 процесса		4 процесса		6 процессов		8 процессов	
	Время, мс	Ускорение	Время, мс	Ускорение	Время, мс	Ускорение	Время, мс	Ускорение
22623	13676	1,65	6638	3,40	4402	5,14	3287	6,88

Как видно из результатов эксперимента, ускорение параллельной программы немного ниже расчётного (таблица 1). Это может быть объяснено б*о*льшими издержками на взаимодействие процессов, чем предполагалось. Тем не менее использование распараллеливания позволило сократить время выделения контуров на изображении.

Библиографический список

1. Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.: Техносфера, 2005. 1072 с.

2. Красильников Н. Н. Цифровая обработка 2D- и 3D-изображений: учебное пособие. СПб.: БХВ-Петербург, 2011. 608 с.

3. Якобовский М. В. Введение в параллельные методы решения задач: учебное пособие / Предисл.: В. А. Садовничий. М.: Изд-во Московского университета, 2013. 328 с.

4. Гергель В. П. Современные языки и технологии параллельного программирования: учебник / Предисл.: В. А. Садовничий. М.: Изд-во Московского университета, 2013. 408 с.

Абрамов Игорь Анатольевич
Пензенский государственный университет, г. Пенза, Россия
E-mail: igora379@gmail.com

Abramov I.A.
Penza State University, Penza, Russia

Банникова Анастасия Николаевна
Пензенский государственный университет, г. Пенза, Россия
E-mail: bannikovaanastasia@yandex.ru

Bannikova A.N.
Penza State University, Penza, Russia

Мартышкина Наталья Александровна
Пензенский государственный университет, г. Пенза, Россия
E-mail: martyskina@yandex.ru

Martyshkina N.A.
Penza State University, Penza, Russia