УДК 681.3 DOI: 10.26102/2310-6018/2019.25.2.011

В. В. Берников, А. П. Преображенский, О.Н. Чопоров

ВОЗМОЖНОСТИ РАСПАРАЛЛЕЛИВАНИЯ ОБРАБОТКИ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ OPENCV И OPENMP

Воронежский институт высоких технологий
Воронежский государственный технический университет,
Воронеж, Россия

Актуальность исследования обусловлена необходимостью решения задач, связанных с обработкой изображений в различных технических приложениях. Рассмотрено несколько подходов: на основе обычного доступа к пикселям изображения, когда фактически осуществляется обход всех значений массива по очереди, доступ к пикселям осуществляется при помощи арифметических операций над указателями, пиксели при этом располагаются внутри одного непрерывного блока памяти последовательным образом, и, на основе предложенного подхода, связанного с распараллеливанием вычислений, использованием многопоточности. На основе эмпирических исследований была показана возможность ускорения вычислений на основе предложенного метода в несколько раз. Рассмотрен блочный алгоритм бинаризации, когда бинарные блоки формируют полное бинарное изображение. В рамках данного алгоритм проведено распараллеливание вычислений. При реализации алгоритма использовался язык C++ и библиотеки OpenCV и OpenMP. На основе эмпирических исследований в Таблицах и на графиках показано, что за счет распараллеливания даже при полной загрузке ядра время обработки изображения было уменьшено почти в 2 раза, что подтверждает возможности применения предлагаемого алгоритма.

Ключевые слова: OpenMP, OpenCV, распараллеливание вычислений, обработка изображений.

Введение

Извлечение информации из растровых документов и их редактирование является трудоемким процессом. Поэтому для редактирования и последующей обработки в системах автоматического проектирования используются векторные форматы файлов. Полутоновое изображение содержит несколько слоев объектов. Каждый слой можно превратить в бинарный и затем его векторизовать с помощью программы - векторизатора. Расслоение связано с получением нескольких бинарных слоев объектов из одного полутонового изображения.

Существует множество алгоритмов, связанных с обработкой бинарных растров: проведение заливки по замкнутым областям растра, проведение утолщения и сужения контуров, построение остова по линиям, построение огибающих контуров и др. Такие алгоритмы представляют основу по всем типам обработки в бинарных растрах и для них характерна

высокая надежность и оперативность. Хуже обстоит дело с надежностью и оперативностью расслоения растра и получения бинарного изображения. В процессе бинаризации существуют искажения таких видов: есть разрывы по символам, текстовым надписям, линиям; размываются изображения линий и текстовым надписям; теряется целостность объектов; появляется шум внутри однородных областей.

Возникновение разрывов и потерь целостности объектов изображений связано с совокупностью причин, которые обусловлены структурой изображений: значительная зональная неравномерность яркости объектов; касание объектов изображений, когда часть контуров объекта будет потеряна при бинаризации, и объект будет сливаться с соседним; идет наложение объектов изображений [1-4].

В данной статье проведены исследования возможности распараллеливания обработки изображений с помощью OPENCV и OPENMP на основе сравнения результатов с обычными последовательными подходами.

1. Реализация параллельного доступа к пикселям в OpenCV

Одним из важнейших пунктов при выполнении алгоритмов обработки изображения является доступ к пикселям изображения или кадра видеофрагмента, в виду того, что таким образом может быть извлечена информация о данном пикселе для последующего успешного выполнения данного алгоритма.

Однако, в любом изображении может содержаться несколько сотен тысяч пикселей и некоторые из методов обработки, которым требуется информация о каждом пикселе изображения для успешного выполнения, могут требовать достаточно много времени для исполнения. Поэтому вопрос быстрого доступа к пикселям является особенно важным на этапе обработки изображения.

Рассмотрим способы доступа и обработки пикселей различными способами с помощью C++ и тестового компьютера, конфигурация которого представлена в Таблице 1. Все способы будут применяться по отношению к единичному изображению, которое будет загружено в многомерный массив типа "Mat", предоставляемый библиотекой OpenCV, и реализующий такие понятия как строка "rows" и "cols" упрощающие доступ соответственно к строкам и столбцам массива.

Таблица 1 – Тестовая конфигурация компьютера

Процессор	1 двухъядерный процессор Intel Core i3-6100 (3.7GHz)

Память	8 Gb
Операционная система	Microsoft Windows 10
Среда разработки	Microsoft Visual Studio Professional 2017 Version 15.7.1

Так же для упрощения доступа к пикселям будет обозначен новый тип переменных Pixel:

```
typedef Point3_<uint8_t> Pixel;
```

Данный тип реализует доступ к значениям x, y и z, которые представляют собой значения красного, зеленого и синего цветов пикселя соответственно.

Тестируемой функцией является функция для закраски пикселя черный в соответствии с условием, если среднее арифметическое значений цветов пикселей будет меньше 255.0, иначе пиксель будет закрашен в белый цвет.

Функция будет выглядеть следующим образом:

```
void colorIt(Pixel& pixel)
{
 const float arMean = (float(pixel.x) + float(pixel.y) +    float(pixel.z)) /
3;
 if (arMean > 255.0)
 {
  pixel.x = 255;
  pixel.y = 255;
  pixel.z = 255;
 }
 else
 {
  pixel.x = 0;
  pixel.y = 0;
  pixel.z = 0;
 }
```

```
}
```

Рассмотрим три основных способа доступа к пикселям:

1. обычный доступ к пикселям с помощью метода "at";
2. доступ к пикселям посредством арифметических операций над указателями;
3. доступ к пикселям с использованием метода "forEach", реализованном в классе "Mat".

Первый способ представляет собой обход всех значений массива по очереди и не использует возможности многопроцессорных систем, что является крайне неэффективным методом еще и в виду того, что расположение значения в массиве каждый раз рассчитывается с нуля при вызове метода "at". Основной цикл данного способа выглядит так:

```
for (int rowNum = 0; rowNum < pixelArray.rows; rowNum++)
{
 for (int colNum = 0; colNum < pixelArray.cols; colNum++)
 {
  Pixel pixel = pixelArray.at<Pixel>(rowNum , colNum);
  colorIt(pixel);
 }
}
```

Второй способ подразумевает под собой прямой доступ к памяти с помощью указателей, минуя сторонние методы. Данный способ осуществим благодаря тому, что пиксели будут расположены в одном непрерывном блоке памяти последовательно. Выглядеть цикл данного способа будет так:

```
Pixel* pixel = pixelsArray.ptr<Pixel>(0,0);
const Pixel* finalPixel = pixel + pixelsArray.cols * pixelsArray.rows;
for (; pixel != finalPixel; pixel++)
{
 colorIt(*pixel);
}
```

Третий способ подразумевает использование метода "forEach" класса "Mat", в который, при вызове, требуется передать функтор. Главным плюсом данного способа является то, что в нем реализовано преимущество

многопоточности и, соответственно, при его выполнении будут нагружаться все доступные мощности системы.

Данный функтор будет выглядеть так:

```
struct FuncObj
{
 void funcObj()(Pixel& pixel, const int* position)
 {
  colorIt(pixel);
 }
};
```

Вызов самого метода "forEach" выглядит так:

```
pixelsArray.forEach<Pixel>(FuncObj());
```

Для тестирования вышеперечисленных способов работы с пикселями было использование изображение с шумом размерами 8000x8000 пикселей, фрагмент изображения приведен на Рисунке 1.

Рисунок 1 — Фрагмент изображения с шумом

Результаты выполнения трех вышеперечисленных способов взаимодействия с пикселями изображения приведены в Таблице 2.

Таблица 2 – Результаты тестирования

Метод	Затраченое время (мс)
at	5866
С помощью указателя	5794
forEach	1721

Таким образом, наблюдается небольшое ускорение обработки пикселей, при прямом доступе к памяти с помощью указателей, по сравнению со стандартным способом с постоянным вызовом метода "at", однако, по сравнению способа "forEach" с другими, укорение достигает 3,4 раза. Так же стоит заметить то, что при увеличении количества ядер процессора, данный способ будет предоставлять еще большее ускорение, в виду своей ориентированности на многопроцессорные системы.

2. Алгоритмы бинаризации

Бинаризацией изображения называется процесс перевода полноцветного или в градациях серого в монохромное изображение. Данный процесс является актуальным для широкого спектра прикладных задач, связанных с анализом и обработкой изображений, в виду того, что позволяет выделить из изображения полезную информацию определенным образом. Большое практическое применение полученные бинаризированные изображения могут найти в цифровой картографии, геоинформационных системах, пространственном анализе, видеонаблюдении, полиграфии и др. [5 - 8].

Бинаризация изображения представлена большим количеством разнообразных алгоритмов, однако, их можно разделить на две группы [9 - 15]: пороговые методы и способы, базирующиеся на условии равенства яркостей.

Пороговые методы требуют нахождения определенной характеристики-порога, с помощью которого можно разделить все представленные пиксели на черные и белые.

Данная пороговая характеристика может быть как статичной, так и адаптивной, для которого выбор значения может основываться на гистограмме значений яркостей. Пример использования данного метода со статичным значением порогового значения представлены на Рисунке 4 и

Рисунке 5. На Рисунке 2 представлено начальное изображение, а на Рисунке 3 полутоновое.

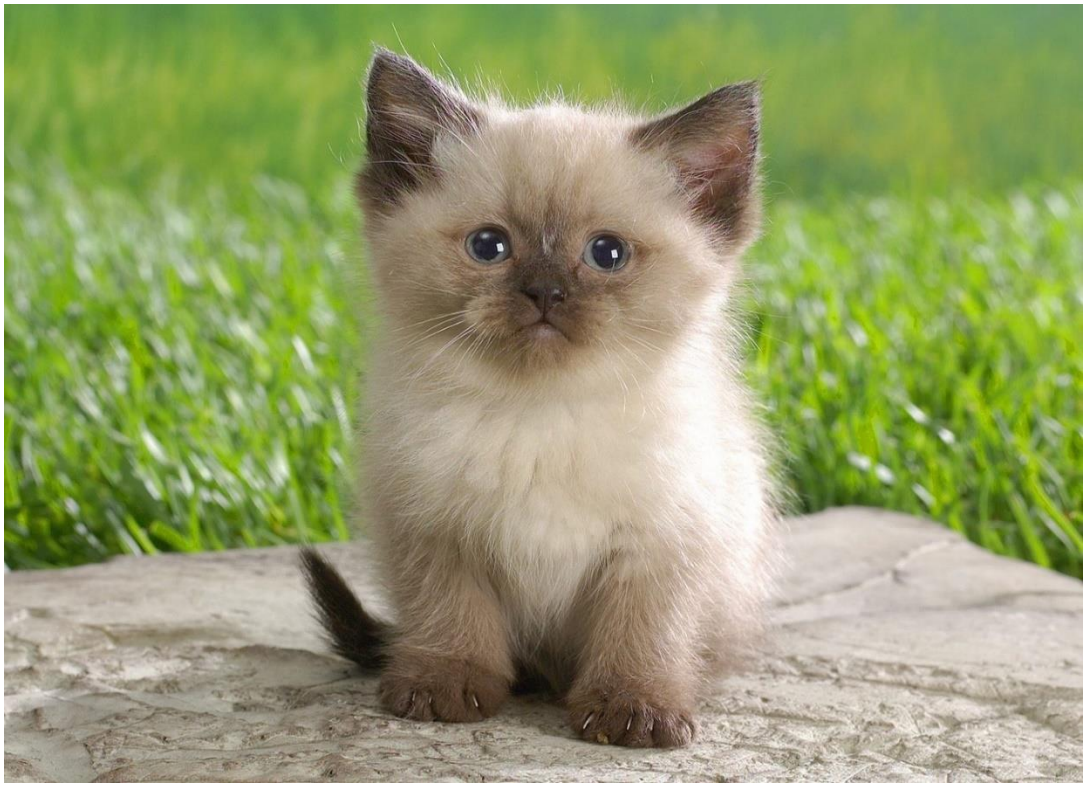

Рисунок 2 — Начальное изображение

Рисунок 3 — Полутоновое изображение

Рисунок 4 – Бинаризированное изображение, порог 100

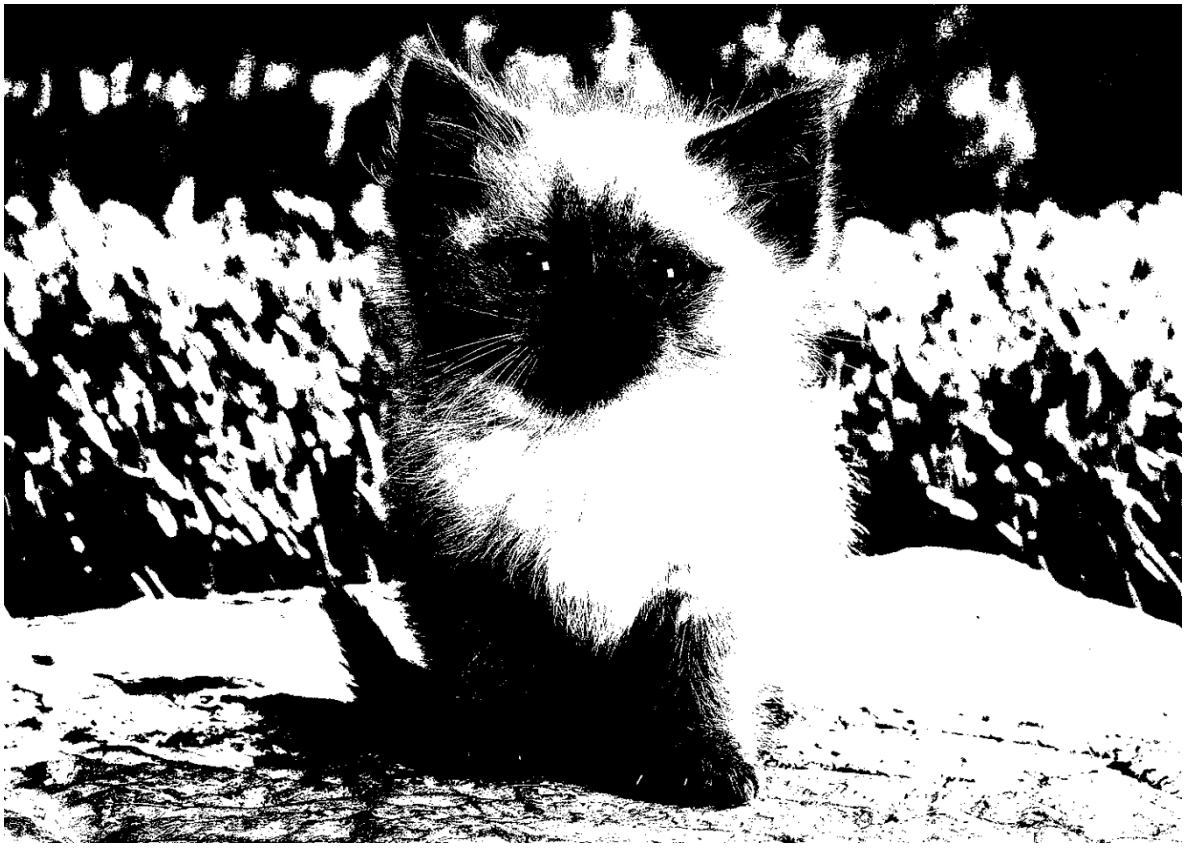

Рисунок 5 – Бинаризированное изображение, порог 150

Методы на основе условия яркостей бинаризируют изображение таким образом, чтобы конечный результат имел такую же яркость, что и изначальное изображение.

Однако, методы расстановки белых и черных пикселей в таком методе могут быть различны.

Они могут быть применены как после решения задачи расстановки, так и используя заранее заготовленные шаблоны с равной яркостью, этот процесс называется псевдотонированием. В таком методе каждому блоку начального изображения ставится в соответствие шаблон с равной яркостью, таким образом различные блоки могут быть закодированы одинаковыми шаблонами.

Пример использования данного метода изображен на Рисунке 6. На Рисунке 2 представлено начальное изображение, а на Рисунке 3 полутоновое.

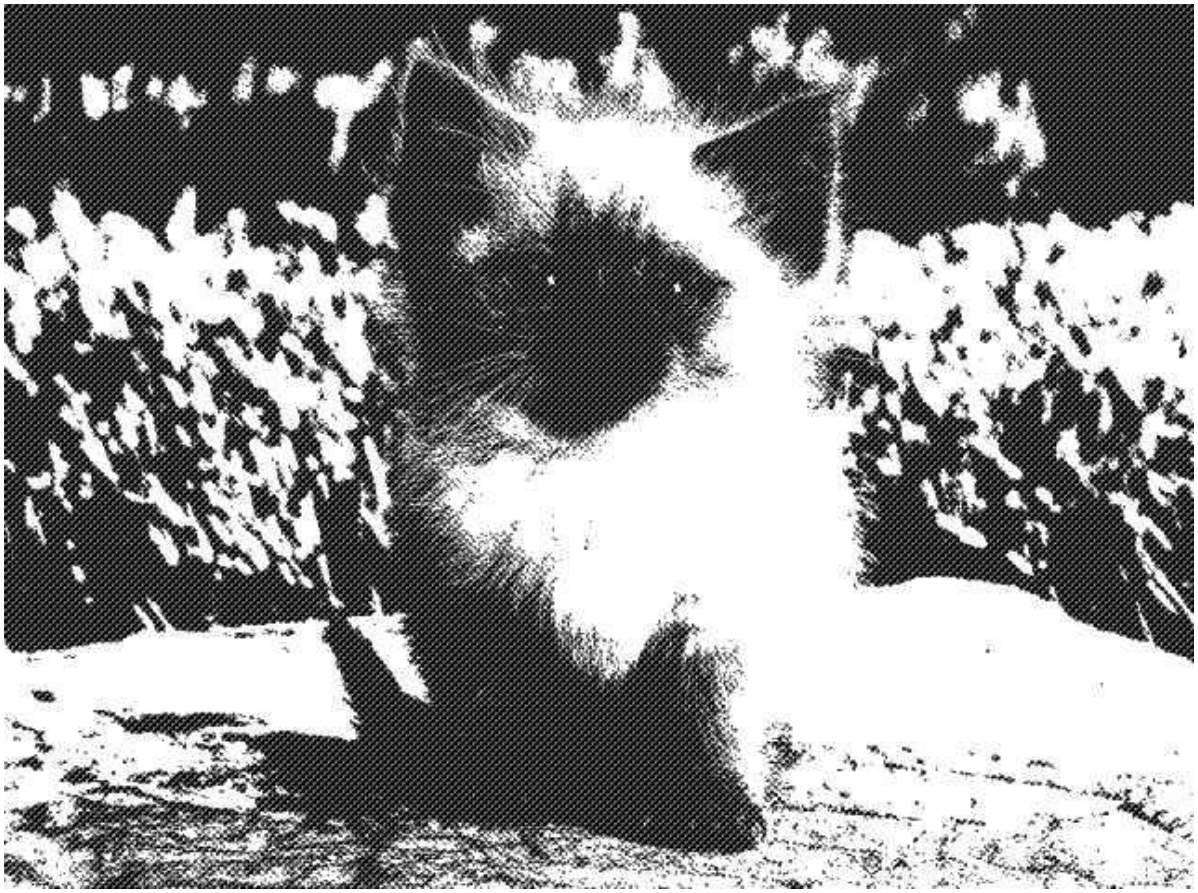

Рисунок 6 – Бинаризированное изображение, метод Байера

4. Реализация алгоритма блочной бинаризации с распараллеливанием

Рассмотрим блочный алгоритм бинаризации [2].

Пусть $f = f(x,y)$, $x = 1..m, y = 1..n$ – исходное изображение с размером $m ? — n$, а $g = g(x,y)$ – изображение, которое получается как результат бинаризации.

Происходит разделение исходного полутонового изображения на N неперекрывающихся блоков f_i, $i = 1..N$. Идет присвоение нулевого значения элементам блока, которые имеют пустое пересечение с исходным изображением. Для соответствующего бинарного блока g_i будет тот же размер, что и f_i, для данного этапа происходит его заполнение нулями. Обработка каждого блока после разбиения происходит независимым образом.

Для каждого блока происходит вычисление суммарной приведённой яркости, которая определяет количество единиц u_i для соответствующего бинарного блока g_i:

$$u_i = \left[2^{-k} ?_{x,y?€€f_i} f(x,y)\right].$$

Для бинарного блоке g_i единицы ставят на позиции, для которых будут находиться u_i наибольших значений яркостей $f(x,y)$ блока f_i, если начинать с максимального значения. Если были найдены несколько одинаковых наибольших значений, тогда выбор одного из нескольких вариантов происходит случайным образом, основываясь на некоторой вероятности.

На основе бинарных блоков g_i происходит формирование полного бинарного изображения.

Данный алгоритм обладает структурой, которую можно распараллелить, в виду того, что обработка каждого из блоков происходит в полной независимости от остальных.

Для реализации данного алгоритма были выбраны: язык C++ с библиотеками OpenCV [17 - 19] и OpenMP [16].

OpenCV – распространяется в условиях лицензии BSD, а так же бесплатна при использовании как в научных целях, так и коммерческих. Имеет реализацию для C, C++, Python, Javai некоторых других языков программирования. Он поддерживает следующие операционные системы: Windows, Linux, MacOS, IOS, Android.

Эта библиотека была спроектирована для эффективного вычисления в приложениях, работающих в реальном времени. Сама библиотека написана с помощью языков Си C++, так же она может использовать преимущества многоядерных систем. Помимо всего этого так же присутствует достаточно большое сообщество по всему миру, помогающие развитию библиотеки.

OpenMP в свою очередь позволяет реализовать параллельные вычисления при помощи многопоточности. В ней «главным» потоком идет создание набора подчиненных потоков и происходит распределение задачи между ними. Мы предполагаем, что исполнение потоков происходит параллельным образом на машине, в которой несколько процессоров, причем количество процессоров не обязательно должно быть большим или равно числу потоков.

Тестирование алгоритма блочной бинаризации с распараллеливанием проводилось на компьютере, конфигурация которого была представлена выше в Таблице 1.

Ввиду наличия технологии Hyper-Threading у данного процессора тестирование проводилось в следующих реализациях:

- одно ядро, один поток;
- одно ядро, два потока;

- два ядра, два потока;
- два ядра, четыре потока.

Результаты тестирования представлены в Таблице 3 и на Рисунке 7. Исходное изображение было увеличено до размера 20000— 16000 пикселей. В дальнейшем будут использованы следующие сокращения 1я – 1 ядро; 1п – 1поток; 2я – 2ядра; 2п – 2потока; 4п – 4потока.

Таблица 3 – Результаты тестирования

ремя работ ы, мс	Длина стороны блока										
				0	2	4	6	8	0	2	4
я, 1п	564	604	680	790	215	398	778	106	400	750	0375
я, 2п	409	490	586	599	803	155	561	988	107	534	948
я, 2п	804	953	025	094	152	238	557	978	035	169	461
я, 4п	701	790	850	936	989	051	320	789	900	000	237

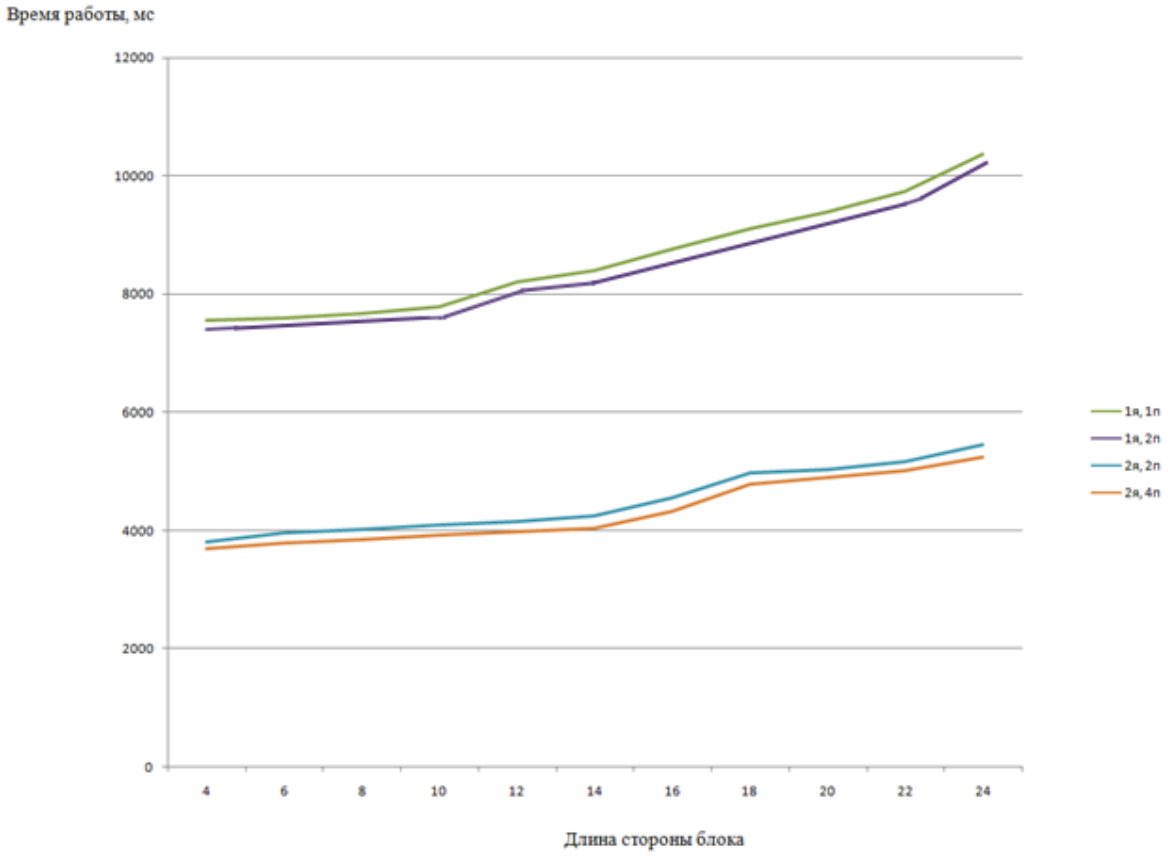

Рисунок 7 – Зависимость времени работы программы от размера блока

На Рисунке 7 видно, что дополнительные потоки, предоставленные технологией Hyper-Threading не позволяют достаточно сильно уменьшить время обработки изображения, из-за загрузки ядра, однако, дополнительное ядро уменьшило время работы почти в 2 раза, что подтверждает достоинство распараллеливания данного алгоритма.

Заключение

В данной статье были рассмотрены основные способы взаимодействия с пикселями изображения с помощью стандартных средств C++ и OpenCV, а также получены опытным путем результаты, обосновывающие приоритет многоядерных систем над одноядерными на начальных этапах обработки изображений.

Был предложен способ распараллеливания вычислений при обработке изображений с помощью реализации в среде OpenMP. Результаты тестирования демонстрируют возможности применения многоядерных систем в ходе многоэтапных обработок изображений.

ЛИТЕРАТУРА

1. Бинаризация черно-белых изображений: состяние и перспективы [Электронный ресурс] / А. Федоров. Режим доступа: http://it-claim.ru/Library/Books/ITS/wwwbook/ist4b/its4/fyodorov.htm
2. Яковлева Е. С., Макаров А. А. О свойствах блочного алгоритма бинаризации цифровых изображений / Е. С.Яковлева, А. А Макаров // Компьютерные инструменты в образовании. — 2015. — № 4. — С. 26–36
3. Dawson-Howe K. A Practical Introduction to Computer Vision with OpenCV. Wiley-IS&T Series in Imaging Science and Technology. — 1 edition. — Wiley, 2014.
4. Гонзалес Р., Вудс Р. Цифровая обработка изображений. / Р.Гонзалес, Р.Вудс // М.: Техносфера, 2006. – 1072 с.
5. GRASS GIS [Электронный ресурс]: Home. Режим доступа: https://grass.osgeo.org/
6. A Digital Image Mapping System [Электронный ресурс]: A software and hardware solution for the various mapping tasks is being developed in a joint project of research institutes and Teragon Context AB. Режим доступа: http://www.isprs.org/proceedings/XXVII/congress/part2/380_XXVII-part2-sup.pdf
7. Juhasz Z. An analytical method for predicting the performance of parallel image processing operations // The Journal of supercomputing. — 1998. — Vol. 12. — P. 157–174.
8. О свойствах блочного алгоритма бинаризации цифровых [Электронный ресурс] / Яковлева Е. С., Макаров А. А. Режим доступа: www.ipo.spb.ru/journal/content/1784/O%20свойствах%20блочного%20алгоритма%20бинаризации%20цифровых%20изображений..pdf
9. Manisha Chate Object Detection and tracking in Video Sequences / Manisha Chate, S.Amudha,Vinaya Gohokar // ACEEE Int. J. on Signal & Image Processing, Vol. 03, No. 01, Jan 2012.
10. Anaswara S Mohan Video Image Processing for Moving Object Detection and Segmentation using Background Subtraction / Anaswara S Mohan, R.Resmi // IEEE International Conference on Computational Systems and Communications (ICCSC), Vol. 01, no. 01, pp.288-292, 17-18 Dec 2014.
11. Weiming Hu. A Survey on Visual Surveillance of Object Motion and Behaviors / Weiming Hu, Tieniu Tan,Liang Wang, and Steve Maybank // IEEE Transactions on systems, man, and cyberneticsapplications and reviews, vol. 34, no. 3, pp. 334- 352, august 2004

12. Asim R. Aldhaheri. Detection and Classification of a Moving Object in a Video Stream / Asim R. Aldhaheri and Eran A. Edirisinghe // In Proc. of the Intl. Conf. on Advances in Computing and Information Technology-ACIT, 2014.
13. Singla M. Motion Detection Based on Frame Difference Method International / M. Singla // Journal of Information & Computation Technology. 2014. Vol. 4. No. 15. P. 1559–1565.
14. Zivkovic Z. Improved adaptive gaussian mixture model for background subtraction / Z. Zivkovic // IEEE Int. Conf. Pattern Recognition. 2004. Vol. 2. P. 28–31
15. Bouwmans T. Background Modeling using Mixture of Gaussians for Foreground Detection – A Survey / T. Bouwmans, F. El Baf, B.Vachon // Recent Patents on Computer Science. 2008. Vol. 1. P. 219– 237.
16. Quinn Michael J. Parallel Programming in C with MPI and OpenMP / J. Quinn Michael // McGraw-Hill Inc, 2004.
17. Learning OpenCV [Электронный ресурс]: Learning OpenCV by Gary Bradski and Adrian Kaehler. Режим доступа: https://www.bogotobogo.com/cplusplus/files/OReilly%20Learning%20OpenCV.pdf
18. Howse J. OpenCV: Computer Vision Projects with Python / J. Howse, P.Joshi, M.Beyeler // United Kingdom, Packt Publishing, 2016. 570 p.
19. Laganiere R. OpenCV 3 Computer Vision Application Programming Cookbook / R. Laganiere // United Kingdom, Packt Publishing, 2017. 474 p.

V. V. Bernikov, A. P. Preobrazhenskiy, O. N. Choporov

THE POSSIBILITY OF PARALLELIZATION OF IMAGE PROCESSING USING OPENCV AND OPENMP

Voronezh institute of high technologies
Voronezh state technical University
Voronezh, Russia

The relevance of the study is due to the need to solve problems related to image processing in various technical applications. Several approaches are considered: on the basis of the usual access to image pixels, when in fact all values of the array are bypassed in turn, access to pixels is carried out using arithmetic operations on pointers, pixels are located within a single continuous memory block in a sequential manner, and, on the basis of the proposed approach associated with parallelization of calculations, using multithreading. On the basis of empirical studies, the possibility of accelerating the calculations based on the proposed method several times was shown. A block binarization algorithm is considered when binary blocks form a complete binary image. Within this algorithm parallelization of calculations is carried out.

When implementing the algorithm, the C++ language and OpenCV and OpenMP libraries were used. On the basis of empirical studies in tables and graphs, it is shown that due to parallelization, even at full load of the kernel, the image processing time was reduced by almost 2 times, which confirms the possibility of using the proposed algorithm.

Keywords: OpenMP, OpenCV, parallel computing, image processing.

REFERENCES

1. Бинаризация черно-белых изображений: состяние и перспективы [Электронный ресурс] / А. Федоров. Режим доступа: http://it-claim.ru/Library/Books/ITS/wwwbook/ist4b/its4/fyodorov.htm
2. Яковлева Е. С., Макаров А. А. О свойствах блочного алгоритма бинаризации цифровых изображений / Е. С.Яковлева, А. А Макаров // Компьютерные инструменты в образовании. — 2015. — № 4. — С. 26–36
3. Dawson-Howe K. A Practical Introduction to Computer Vision with OpenCV. Wiley-IS&T Series in Imaging Science and Technology. — 1 edition. — Wiley, 2014.
4. Гонзалес Р., Вудс Р. Цифровая обработка изображений. / Р.Гонзалес, Р.Вудс // М.: Техносфера, 2006. – 1072 с.
5. GRASS GIS [Электронный ресурс]: Home. Режим доступа: https://grass.osgeo.org/
6. A Digital Image Mapping System [Электронный ресурс]: A software and hardware solution for the various mapping tasks is being developed in a joint project of research institutes and Teragon Context AB. Режим доступа: http://www.isprs.org/proceedings/XXVII/congress/part2/380_XXVII-part2-sup.pdf
7. Juhasz Z. An analytical method for predicting the performance of parallel image processing operations // The Journal of supercomputing. — 1998. — Vol. 12. — P. 157–174.
8. О свойствах блочного алгоритма бинаризации цифровых [Электронный ресурс] / Яковлева Е. С., Макаров А. А. Режим доступа: www.ipo.spb.ru/journal/content/1784/O%20свойствах%20блочного%20алгоритма%20бинаризации%20цифровых%20изображений.pdf
9. Manisha Chate Object Detection and tracking in Video Sequences / Manisha Chate, S. Amudha, Vinaya Gohokar // ACEEE Int. J. on Signal & Image Processing, Vol. 03, No. 01, Jan 2012.
10. Anaswara S Mohan Video Image Processing for Moving Object Detection and Segmentation using Background Subtraction / Anaswara S Mohan, R. Resmi // IEEE International Conference on Computational Systems and Communications (ICCSC), Vol. 01, no. 01, pp.288-292, 17-18 Dec 2014.

11. Weiming Hu. A Survey on Visual Surveillance of Object Motion and Behaviors / Weiming Hu, Tieniu Tan, Liang Wang, and Steve Maybank // IEEE Transactions on systems, man, and cyberneticsapplications and reviews, vol. 34, no. 3, pp. 334- 352, august 2004
12. Asim R. Aldhaheri. Detection and Classification of a Moving Object in a Video Stream / Asim R. Aldhaheri and Eran A. Edirisinghe // In Proc. of the Intl. Conf. on Advances in Computing and Information Technology-ACIT, 2014.
13. Singla M. Motion Detection Based on Frame Difference Method International / M. Singla // Journal of Information & Computation Technology. 2014. Vol. 4. No. 15. P. 1559–1565.
14. Zivkovic Z. Improved adaptive gaussian mixture model for background subtraction / Z. Zivkovic // IEEE Int. Conf. Pattern Recognition. 2004. Vol. 2. P. 28–31
15. Bouwmans T. Background Modeling using Mixture of Gaussians for Foreground Detection – A Survey / T. Bouwmans, F. El Baf, B. Vachon // Recent Patents on Computer Science. 2008. Vol. 1. P. 219– 237.
16. Quinn Michael J. Parallel Programming in C with MPI and OpenMP / J. Quinn Michael // McGraw-Hill Inc, 2004.
17. Learning OpenCV [Электронный ресурс]: Learning OpenCV by Gary Bradski and Adrian Kaehler. Режим доступа: https://www.bogotobogo.com/cplusplus/files/OReilly%20Learning%20OpenCV.pdf
18. Howse J. OpenCV: Computer Vision Projects with Python / J. Howse, P. Joshi, M. Beyeler // United Kingdom, Packt Publishing, 2016. 570 p.
19. Laganiere R. OpenCV 3 Computer Vision Application Programming Cookbook / R. Laganiere // United Kingdom, Packt Publishing, 2017. 474 p.