Исследование параллельных CUDA- реализаций алгоритма построения карты диспарантности по разноракурсным изображениям*

А.П. Котов[1,2], В.А. Фурсов[1,2], Е.В. Гошин[1,2]

Самарский государственный аэрокосмический университет
им. академика С.П.Королева[1], Институт систем обработки изображений РАН[2]

В статье решается задача повышения быстродействия алгоритма формирования по разноракурсным изображениям карты диспарантности, по которой затем строится модель трехмерной сцены. Основным наиболее затратным этапом алгоритма является определение относительных сдвигов точек на разноракурсных изображениях. Ранее авторами был разработан эффективный алгоритм построения карт диспарантности [1], в котором для повышения быстродействия и надежности используется процедура формирования пирамиды изображений. Настоящая работа посвящена исследованию быстродействия и эффективности реализации соответствующего параллельного алгоритма в CUDA-среде.

Ключевые слова: цифровая обработка изображений, реконструкция 3D-сцен по разноракурсным изображениям, сопоставление изображений, CUDA-технология.

1. Постановка задачи

В работе исследуется возможность повышения быстродействия технологии построения трехмерных моделей сцен по парам разноракурсных изображений. Необходимость этого исследования мотивируется тем, что часто эта задача должна решаться в реальном времени. Например, это требование является необходимым при обработке данных ДЗЗ с целью оперативного анализа чрезвычайных ситуаций, террористических угроз и др. Таким образом, актуальной является задача построения быстродействующих и в то же время относительно недорогих и экономичных, по требуемым вычислительным ресурсам, вычислительных технологий восстановления 3D-сцен по разноракурсным изображениям.

Для определения трехмерных координат точек сцены используются координаты соответствующих точек на двух изображениях одной и той же точки сцены. Обычно полагают, что задача восстановления трехмерной сцены решена, если построена так называемая карта диспарантности, которая представляет собой поле значений относительных сдвигов соответствующих точек на двух разноракурсных изображениях. В технологиях восстановления трехмерных сцен по разноракурсным изображениям нахождение соответствующих точек на разных видах является основной вычислительной проблемой. Трудности существенно возрастают, если изображения имеют большие относительные сдвиги. При этом область поиска соответствующих точек на изображениях также должна быть увеличена, что сопровождается значительным увеличением времени поиска и уменьшением надежности их определения.

Эту трудность обычно преодолевают путем ректификации изображений (приведение строк изображений к одинаковой ориентации). Однако при этом вносятся дополнительные искажения, связанные с интерполяцией отсчетов изображений [2]. В работе [3] авторами была предложена технология, не требующая предварительной ректификации изображений, в которой соответствующие точки ищутся на соответствующих эпиполярных линях. Однако при больших начальных относительных сдвигах и различиях масштабов сопоставляемых изображений область остается достаточно большой.

В работе [1] для преодоления указанных проблем предложено ввести этап начального совмещения разноракурсных изображений. В настоящей работе эта идея реализована в CUDA среде на гибридных вычислительных устройствах, включающих графические процессоры. Общая

* Работа выполнена при поддержке Министерства образования и науки Российской Федерации.

схема основных этапов предложеной технологии реконструкции 3D-сцены по разноракурсным изображениям приведена на рисунке 1.

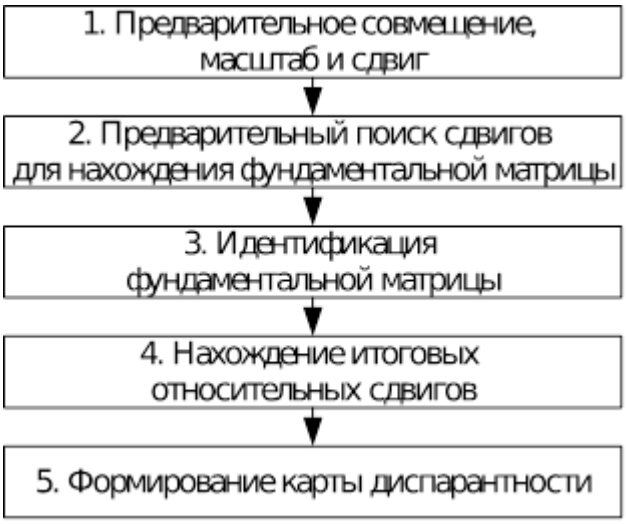


Рис. 1. Схема этапов технологии

Введение дополнительного этапа начального приближенного совмещения разноракурсных изображений позволяет существенно повысить быстродействие этапа сопоставления с использованием «пирамиды изображений». Алгоритм пирамиды изображений строится в виде иерархической схемы вычислений. Пирамида изображений формируется в виде набора изображений, получаемых уменьшением разрешения в два раза по обеим координатам так, что на N-м уровне пирамиды формируется изображение, разрешение которого в 2^N раз меньше исходного разрешения.

Число уровней пирамиды задается в виде параметра. На первом шаге данного этапа обрабатывается изображение с наименьшим разрешением и начальным сдвигом равным нулю. На следующем шаге используется информация о сдвиге, найденном на предшествующем шаге так, что на каждом следующем шаге значения координат удваиваются. Найденные таким образом соответствующие точки используются для определения фундаментальной матрицы.

На следующем этапе осуществляется поиск соответствующих точек для построения карты диспарантности. Используется метод, основанный на использовании в качестве штрафного коэффициента в минимизируемой функции расстояния до эпиполярной линии [4]. Существо метода сводится к следующему.

Пусть координаты точек на первом изображении (u,v), а координаты соответствующих им точек на втором – $(u+\Delta u, v+\Delta v)$, где Δu, Δv – относительные сдвиги координат u, v соответственно, а

$$I(u,v),\quad I'(u+\Delta u, v+\Delta v) \tag{1}$$

– функции распределения яркости отсчетов на этих изображениях.

Задача состоит в поиске для каждой точки (u,v) на первом изображении соответствующей точки $(u+\Delta u, v+\Delta v)$ на втором изображении посредством минимизации критерия сходства:

$$E(u_0,v_0,\Delta u,\Delta v) = \\ = \sum_{(u,v)\in D(u_0,v_0)} a(u,v)\left\|I(u,v)-I'(u+\Delta u,v+\Delta v)\right\|, \tag{2}$$

где $D(u_0,v_0)$ – заданная область вокруг точки (u_0,v_0), а $a(u,v)$ – весовая функция, задаваемая в указанной области в виде произведения трех коэффициентов:

$$a(u,v) = w_c \cdot w_d \cdot w_f,$$

где

$$w_d = \exp\left\{-\left\|(u_0,v_0)-(u,v)\right\|^2\right\}, \tag{3}$$

$$w_d = \exp\left\{-\left\|I(u_0,v_0) - I'(u,v)\right\|^2\right\}, \quad (4)$$

$$w_d = \exp\left\{-\frac{au' + bv' + c}{\sqrt{a^2 + b^2}}\right\} \quad (5)$$

Коэффициент w_f имеет смысл функции «штрафа» при удалении точки (u',v') от эпиполярной линии $au' + bv' + c$, определяемой по координатам точки (u_0,v_0) с использованием фундаментальной матрицы F [11,12]:

$a = u_0F_{11} + v_0F_{12} + F_{13}$,

$b = u_0F_{21} + v_0F_{22} + F_{23}$,

$c = u_0F_{31} + v_0F_{32} + F_{33}$.

С использованием полученных значений относительных сдвигов соответствующих точек на разноракурсных изображениях формируется карта диспарантности (этап 5).

Наибольшие вычислительные затраты в приведенной общей схеме алгоритма имеют место на этапе 2 – предварительный поиск сдвигов для нахождения фундаментальной матрицы и на этапе 4 – нахождение итоговых относительных сдвигов. Эти этапы обладают большим внутренним параллелизмом. В частности, нахождение относительных сдвигов для каждой точки (x,y) можно выполнять независимо. При этом количество параллельных процессов равно произведению числа пикселей изображения и числа всех возможных сдвигов (u,v) в области поиска D_2.

В настоящей работе проводится исследование двух вариантов распараллеливания, различающихся числом дескрипторов, вычисляемых в каждой отдельной нити и, как следствие, накладными расходами на пересылки.

2. Описание вариантов параллельного алгоритма

Варианты параллельного алгоритма реализуются на втором и четвертом этапах сквозной технология построения карты диспарантности, показанной на рис.1. Отличие четвертого этапа от второго заключается в использовании штрафного коэффициента для уточнения относительных сдвигов. Использование штрафного коэффициента практически не влияет на вычислительную сложность алгоритма. Поэтому, построение параллельного алгоритма рассмотрено на примере второго этапа.

Под первым вариантом параллельного алгоритма будем понимать параллельную CUDA-реализацию алгоритма, которая использовалась в статье [1]. На рисунке 2 приведена структурная схема этой CUDA-реализации.

Перед запуском CUDA-ядра копируются необходимые данные из оперативной памяти в память (global memory) графической видеокарты. Для результатов также выделяется память на видеокарте. Каждая нить рассчитывает одну евклидову норму для двух выбранных дескрипторов. Под дескриптором данной точки понимается вектор признаков фрагмента изображения с центром в искомой соответствующей точке. Для нахождения соответствующей точки, необходимо рассчитать евклидовы нормы для всех дескрипторов точек, выбранных в области поиска в качестве возможных соответствующих. При этом число создаваемых нитей равно произведению размера изображения (в пикселах) и размера области поиска (также в пикселах).

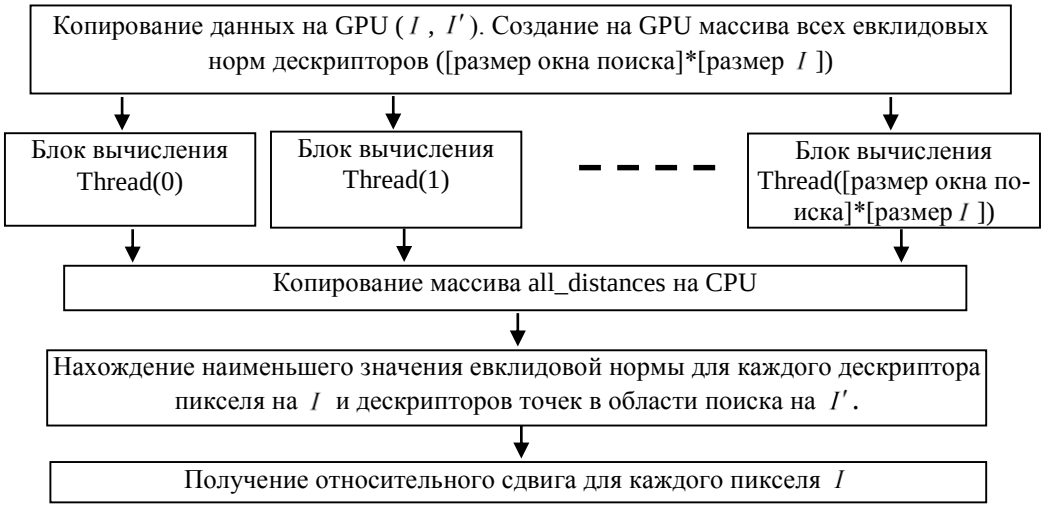


Рис. 2. Структурная схема первого варианта параллельного алгоритма

На рисунке 3 приведена укрупненная схема вычисления, выполняемых одной нитью. Данный блок вычислений обозначен на рисунке 2 обозначен как «Блок вычислений Thread». Каждая нить выполняет вычисления независимо от остальных нитей. Данные, которые используются для вычислений, находятся в памяти видеокарты (global memory). Таким образом, результатом работы нити является евклидова норма для двух выбранных дескрипторов.

Рис. 3. Схема вычислений, выполняемых одной нитью в первом варианте алгоритма

Недостатком первого варианта параллельного алгоритма является добавление этапа выбора минимальной евклидовой нормы вектора дескрипторов. Данный этап выполняется последовательно на CPU. При этом предварительно копируются данные массива с вычисленными евклидовыми нормами из видеопамяти. После определения минимальной нормы на CPU определяется относительный сдвиг.

Структурная схема второго варианта параллельного алгоритма представлена на рисунке 4.

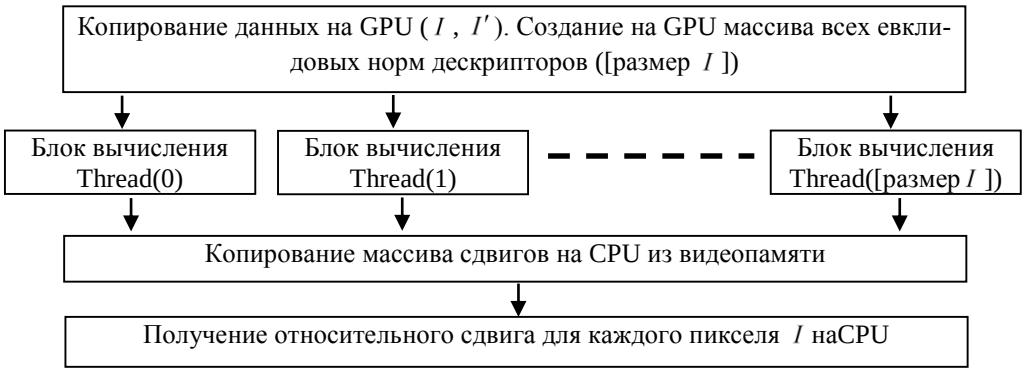


Рис. 4. Структурная схема второго варианта параллельного алгоритма

В этом варианте (в отличие от предыдущего) удается избежать формирования массива избыточных данных на GPU реализации. Также отсутствует последовательный этап сравнений евклидовых норм на CPU.

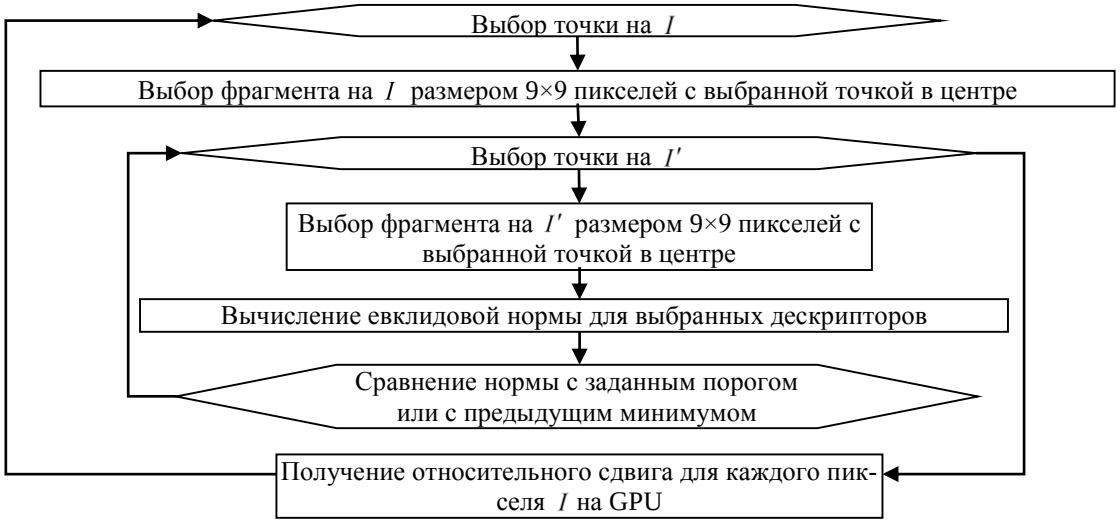


Рис. 5. Схема вычислений, выполняемых одной нитью во втором варианте алгоритма

Предлагаемые параллельные реализации отличаются степенью параллелизма. Во второй CUDA-реализации используется более высокий уровень распараллеливания. Нить, во втором варианте параллельного алгоритма, осуществляет больший объем вычислений, чем нить в первом варианте. На рисунке 5 приведена укрупненная схема блока вычислений, которые выполняются одной нитью.

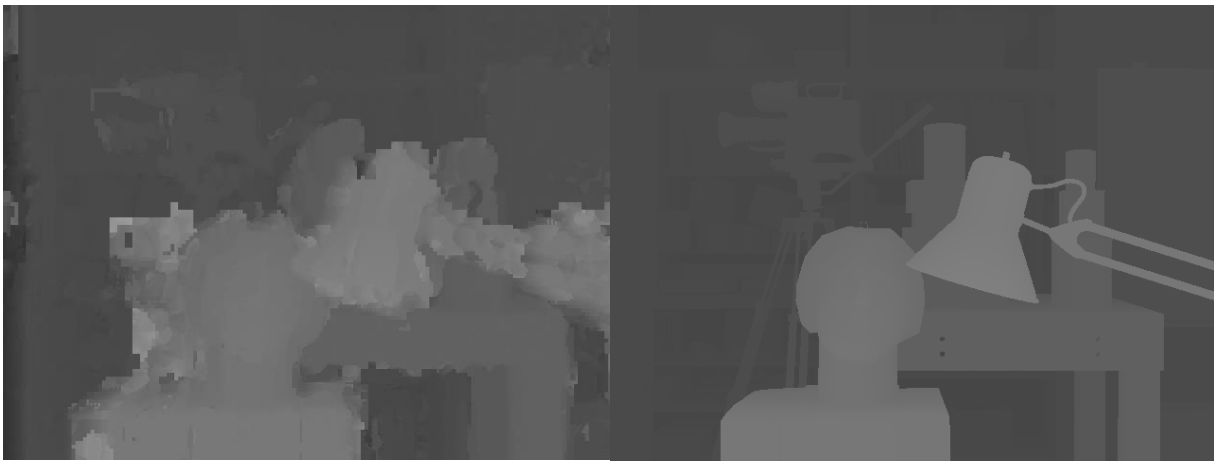
3. Результаты экспериментов

При проведении экспериментов использовались стереоизображения из набора изображений «Tsukuba», которые часто используются в качестве тестовых в задаче сопоставления изображений. Этот выбор был продиктован тем, что указанная база изображений содержит также эталонные карты диспарантности, по которым возможно сопоставление результатов. Исходные изображения представлены на рисунке 6.

а) *б)*

Рис. 6. Исходные изображения «Tsukuba»

С помощью предлагаемой технологии для указанных изображений была сформирована карта диспарантности (рис. 7, а). На рисунке 7, б) для сравнения приведена эталонная карта диспарантности, рассчитанная по тем же изображениям с использованием априорной информации о параметрах камер и хранящаяся в наборе изображений «Tsukuba».



а) *б)*

Рис. 7. Вычисленная (а), и эталонная (б) карты диспарантности

В таблице 1 приведены результаты сравнительных исследований времени реализации алгоритма на CPU и GPU при различном числе уровней пирамиды, задаваемых на первом этапе устранения больших относительных сдвигов (данные получены при реализации с использованием GeForce GTX 750 Ti, и Intel Core i7-6700K, 16GB DDR4, ОС Windows 10).

Таблица 1. Полученные результаты исследования

Уровень пирамиды (разрешение изображения в пикселях)	1 (20×15)	2 (40×30)	3 (80×60)	4(160×120)	5(320×240)	6(640×480)
Время выполнения на CPU (мс)	5.3	24.4	105.2	427.4	1722	6943.5
Время выполнения на GPU параллельной реализации №1 (мс)	2.3	4.3	9.5	36.9	117.1	510.4
Время выполнения этапа сравнения (мс)	0.02	0.12	0.49	2.1	7.4	29.8
Время выполнения параллельной реализации №1 и этапа сравнения (мс)	2.32	4.42	9.99	39	124.5	540.2
Ускорение (с учетом этапа сравнения)	2.28	5.52	10.53	10.96	13.83	12.85
Время выполнения на GPU параллельной реализации №2 (мс)	9.2	9.7	10.3	27.4	110.4	404.2
Ускорение	0.576	2.51	10.21	15.6	15.6	17.18

В качестве исходных данных использовались изображения размером 640×480 пикселей. В таблице 1 приведены результаты для различных размеров изображений: 20×15, 40×30, 80×60, 160×120, 320×240 и 640×480 пикселей.

Результаты для последовательного алгоритма и первого варианта параллельного алгоритма отличаются от результатов приведенного в работе [1]. Связано это с использованием другой вычислительной системы. Кроме того, в статье [1] не учитывался этап сравнения евклидовых норм дескрипторов.

Время выполнения второго параллельного алгоритма больше, чем время выполнения последовательного для изображений размером 20×15 пикселей. Однако, для остальных изображений пирамиды получено ускорение второго параллельного алгоритма относительно последовательного. Было установлено, что время выполнения второго варианта параллельного алгоритма изображений размером 20×15, 40×30 и 80×60 пикселей примерно одинаковое. Это означает, что ресурсы видеокарты используются неэффективно. Однако, для изображений размером 160×120, 320×240 и 640×480 пикселей получены примерно одинаковые значения ускорения второго варианта параллельного алгоритма относительно последовательного, более чем в 15 раз. Тот факт,

что ускорение для разных изображений остается практически неизменным означает эффективное использование ресурсов-мощностей видеокарты (CUDA occupancy).

Время выполнения первого варианта параллельного алгоритма меньше времени выполнения второго для изображений размером 20×15 и 40×30 пикселей. Для изображений размером 80×60 пикселей время выполнения двух вариантов параллельного алгоритма практически одинаково. Однако, время обработки с использованием второго варианта для изображений размером 160×120, 320×240 и 640×480 пикселей, оказалось меньше, чем с использованием первого варианта.

4. Заключение

Результаты исследований показали, что использование первой параллельной CUDA-реализации алгоритма [1] позволяет получить ускорение по сравнению с последовательной реализацией более чем в 12 раз. Однако, из-за избыточного хранения данных на GPU, применение данного подхода на больших изображениях, например, космических снимках, имеет ограничения, связанные с размером памяти видеокарты. Кроме того, приходится дополнительно хранить и обрабатывать больший объем информации в оперативной памяти. Использование второго варианта параллельного алгоритма (рисунок 4) позволило достичь ускорения в 17 раз по сравнению с последовательным алгоритмом для изображений размером 640×480, и в 1,5 раза по сравнению с первым вариантом параллельного алгоритма.

Литература

1. Котов А.П., Фурсов В.А., Гошин Е.В. Технология оперативной реконструкции трехмерных сцен по разноракурсным изображениям // Компьютерная оптика. 2015. Т. 39, № 4. С. 600-605.
2. Hartley R.I. Theory and Practice of Projective Rectification // International Journal of Computer Vision. 1999. Vol. 35. P. 115-127.
3. Фурсов В.А., Гошин Е.В., Бибиков С.А. Реконструкция 3D-сцен на пучках эпиполярных плоскостей стереоизображений // Мехатроника, автоматизация, управление. 2013. №9 (150). С. 19-24.
4. Фурсов В.А., Гошин Е.В. Информационная технология реконструкции цифровой модели местности по стереоизображениям // Компьютерная оптика. 2014. Т. 38. №. 2 С. 335-342.
5. Lowe D.G. Object recognition from local scale-invariant features //Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Ieee, 1999. Т. 2. С. 1150-1157.
6. Bay H. Speeded-up robust features (SURF) / Bay H., Ess A., Tuytelaars T., Van Gool L. //Computer vision and image understanding. 2008. Т. 110. №. 3. С. 346-359.
7. Фурсов В.А., Гошин Е.В. Метод согласованной идентификации в задаче определения соответственных точек на изображениях // Компьютерная оптика. 2012. Том 36 №1 С. 131-135.
8. Фурсов В.А., Гошин Е.В. Решение задачи автокалибровки камеры с использованием метода согласованной идентификации // Компьютерная оптика. 2012. Т. 36. № 4 С. 605-610.
9. Harris C. A combined corner and edge detector / Harris C., Stephens M. //Alvey vision conference. 1988. Т. 15. С. 50.
10. Tao M. SimpleFlow: A Non-iterative, Sublinear Optical Flow Algorithm / M Tao, J Bai, P Kohli, S Paris// Computer Graphics Forum. 2012. Vol. 31. No. 2. P. 345-353.
11. Форсайт Д., Понс Ж. Компьютерное зрение. Современный подход / М.: Издательский дом "Вильямс", 2004. 928 с.
12. Грузман И.С. Цифровая обработка изображений в информационных системах: Учебное пособие / И.С. Грузман, В.С. Киричук, В.П. Косых и др. - Новосибирск: Изд-во НГТУ, 2002. - 352с.

Research of parallel CUDA implementations of the algorithm for constructing the disparity map from stereo images

A.P. Kotov[1,2], V.A. Fursov[1,2], Ye.V. Goshin[1,2]

S.P. Korolyov Samara State Aerospace University[1], Image Processing Systems Institute of the RAS[2]

This paper solves the problem of increasing the efficiency of the algorithm for the disparity map formation from stereo images. A three-dimensional model of the scene is reconstructed using this disparity map. The most computationally complex stage of the algorithm is determination of the relative points' shifts in stereo images. In previous paper we proposed an efficient algorithm for disparity map formation in which a pyramid of images was formed to improve the efficiency and reliability. This paper is dedicated to the study of the efficiency of the corresponding parallel algorithm implementation in CUDA environment.

Keywords: digital image processing, 3D-scene reconstruction, image matching, CUDA.

References

1. Kotov AP, Fursov VA, Goshin YeV Technology for fast 3d-scene reconstruction from stereo images // Computer optics. 2014. Vol 39 №4 P. 600-605.
2. Hartley R.I. Theory and Practice of Projective Rectification // International Journal of Computer Vision. 1999. Vol. 35. P. 115-127.
3. Fursov VA, Goshin YeV, Bibikov SA. 3D-scene stereo reconstruction on sheaves of epipolar planes // Mechatronics automation control 2013. №. 9(150). P. 19-24.
4. Fursov VA, Goshin, YeV. Information technology for digital terrain model reconstruction from stereo images // Computer optics. 2014. Vol 38 №2 P. 335-342.
5. Lowe DG. Object recognition from local scale-invariant features //Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Ieee, 1999. Т. 2. С. 1150-1157.
6. Bay H, Ess A, Tuytelaars T, Van Gool L. Speeded-up robust features (SURF). Computer vision and image understanding. 2008. Vol. 110. №. 3. P. 346-359.
7. Fursov VA., Goshin, YeV. Conformed identification in corresponding points detection problem [In Russian]. Computer optics 2012; Vol 36 №1 p. 131-135.
8. Fursov VA., Goshin, YeV. Solving a camera autocalibration problem with a conformed identification method [In Russian]. Computer optics 2014. Vol 36 №4 P. 605-610.
9. Harris C., Stephens M. A combined corner and edge detector. Alvey vision conference. 1988. Vol. 15. С. 50.
10. Tao M, Bai J, Kohli P, Paris S. SimpleFlow: A Non-iterative, Sublinear Optical Flow Algorithm. Computer Graphics Forum. 2012. Vol. 31. No. 2. P. 345-353.
11. Forsyth D, Ponce J**.** Computer Vision: A Modern Approach. Moscow.: "Williams" Publisher, 2004. 928 p.
12. Gruzman IS. et al. Digital image processing in information systems. NGTU, Novosibirsk. 2002. 352 p.