

ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ РАСПРЕДЕЛЕННЫХ СИСТЕМ КОМПРЕССИИ ВИДЕОИНФОРМАЦИИ

Розроблено модифікований блочний метод оцінки руху в відеопослідовностях. Виконано розпаралелювання алгоритму, що реалізує розроблений метод на графічному процесорі. Експериментально досліджено переваги та недоліки представлених методу та алгоритму. Запропонований метод дозволяє виконувати компресію відеоінформації стандарту високої чіткості в реальному часі без значної втрати якості результату.

The modified block method of motion estimation in video sequences is developed. The parallelization of the algorithm that implements the developed method on a graphical processor is done. Advantages and disadvantages of the presented method and the algorithm are experimentally investigated. The presented method allows compression of high definition video in real time without significant result quality loss.

Введение

Современные стандарты компрессии видеоинформации, такие как, *MPEG-2* [1] и *MPEG-4/H.264* [2] предполагают сжатие видеоинформации за счет устранения ее избыточности, в частности, интракадровой [3] и интеркадровой [4]. Вычислительная сложность методов устранения интеркадровой избыточности высока и занимает до 90% в структуре вычислительной сложности всего процесса компрессии видеоинформации [5], что затрудняет сжатие видеоинформации стандарта высокой четкости (*HD*) [6] в распределенных системах в реальном времени без применения специальных вычислительных средств.

Постановка задачи

Решение задачи оценки движения (ОД) блоков кадров в видеопоследовательностях очень важно для эффективного устранения интеркадровой избыточности видеоинформации, однако, применяемые методы решения этой задачи обладают высокой вычислительной сложностью.

В данной работе ставится задача разработать модификацию базового блочного метода ОД в видеопоследовательностях, обеспечивающую уменьшение времени сжатия, а так же разработать соответствующий алгоритм и его реализацию для графического процессора (ГП или *GPU*), которые позволяют сжимать видеоинформацию стандарта высокой четкости в рас-

пределенных системах в реальном времени без существенной потери качества результата.

1. Организация процесса компрессии видеоинформации

Распределенная система компрессии видеоинформации предполагает наличие анализатора видеопоследовательностей [7], предназначенного для разделения видеоинформации на части с целью последующего распределения этих частей по вычислительным узлам. Получаемые в результате разделения части видеопоследовательностей по мере появления становятся в очередь на сжатие вычислительными узлами. Если какой-либо вычислительный узел свободен, то он изымает видеопоследовательность из очереди и сжимает ее. Сжатая часть видеоинформации поступает в хранилище. Если все ресурсы заняты, то видеопоследовательность ожидает в очереди, пока не освободится хотя бы один процессор.

Такая организация вычислительного процесса позволяет избежать накладных расходов на синхронизацию процессоров, а так же сжимать части видеопоследовательностей по мере их появления. Однако, производительности современных распределенных систем компрессии видеоинформации не достаточно для того, чтобы сжимать при помощи базового метода ОД видеоинформацию, представленную по стандарту высокой четкости, в реальном времени без существенной потери качества [8]. Многие

современные вычислительные узлы распределенных систем имеют векторные сопроцессоры (GPU), которые можно задействовать для распараллеливания вычислений в пределах узла. Для достижения поставленной цели требуется распараллеливание процесса компрессии видеоинформации на каждом узле распределенной системы, которое можно осуществить путем модификации методов оценки движения в видеопоследовательностях и использования специальных вычислительных средств.

2. Разработка модифицированного блочного метода ОД в видеопоследовательностях

Блочные методы ОД разбивают кадры на блоки (рис. 1), далее в текущем кадре для каждого блока C_n выполняется поиск в восстановленном кадре наиболее "похожего" блока C_k , смещенного на вектор движения $v = (v_x, v_y)$. Базовый метод предполагает поиск с фиксированным размером области поиска Ω "похожих" блоков в восстановленном кадре и обладает высокой вычислительной сложностью [9]. Область поиска в алгоритмах "быстрой" ОД существенно сокращается, но такие алгоритмы не гарантируют точное нахождение всех векторов движения [10].

В данной работе предлагается модификация базового блочного метода ОД, основанная на динамическом сокращении области поиска векторов движения для каждого блока и обеспечивающая, таким образом, уменьшение времени сжатия видеопоследовательностей по сравнению с базовым методом.

2.1 Функция "скорости" движения блоков кадров видеопоследовательности

Видеопоследовательности, сформированные анализатором видеоинформации в соответствии с предложенным в [7] методом, характеризуются сильной корреляционной зависимостью между соседними кадрами и, как правило, короткими по продолжительности. "Скорость" движения блоков кадра между смежными кадрами в таких видеопоследовательностях изменяется незначительно. Поэтому начальное приближение для поиска векторов движения $ME(v)$ для каждого блока кадра целесообразно выбрать в соответствии со значением функции "скорости" движения блоков. Затем, уточнить вектор движения по соседним блокам.

Весь кадр F площадью S_F нацело разобьем на N блоков C , равных по площади S_C .

Поскольку частота смены кадров в видеопоследовательности фиксирована, то значение "скорости" движения блоков кадра между смежными кадрами можно вычислить при помощи следующего выражения:

$$MS(C_n, C_k) = \sqrt{(C_n^x - C_k^x)^2 + (C_n^y - C_k^y)^2}, \quad (1)$$

где n – номер текущего кадра; k – номер восстановленного кадра; C_n^x, C_n^y – координаты x и y , соответственно, блока C_n на плоскости разбиения текущего кадра; C_k^x, C_k^y – координаты x и y , соответственно, блока C_k на плоскости разбиения восстановленного кадра.

Выражение (1) будем называть функцией "скорости" движения блоков кадра между смежными кадрами видеопоследовательности.

Оценку движения для каждого блока между первой парой кадров выполним интеркадровым блочным методом с помощью поиска в области фиксированного размера $\Omega = 128$, поскольку, как правило, при такой области поиска вероятность нахождения лучших векторов движения очень высокая. Однако, время, затрачиваемое кодером для выполнения операции ОД очень велико [8]. Межкадровое движение для каждого блока в смежных кадрах оценим при помощи функции "скорости" движения блоков. Для всех последующих кадров начальный блок C_k в восстановленном кадре, относительно которого вычисляем функцию ОД для блока C_n в текущем кадре, можно вычислять на основе уже найденного вектора движения $ME(v)$ для соответствующего блока C_{k-1} предыдущего восстановленного кадра со смещением на расстояние, "пройденное" блоком за один кадр с определенной "скоростью" движения (рис. 2). Поскольку движение блоков кадров происходит с непостоянной скоростью (в зависимости от типа движения объектов в смежных кадрах), то предсказанное положение начального блока для поиска вектора движения может отличаться от реального положения блока, дающего лучший из возможных вектор движения.

Рассмотрим 3 типа движения объектов в смежных кадрах видеопоследовательности:

1. Камера статична, объекты движутся: выпуски телепередач, новостей, мультипликация;

2. Камера движется, объекты статичны: специализированная съемка;
3. Камера движется, объекты движутся: кино.

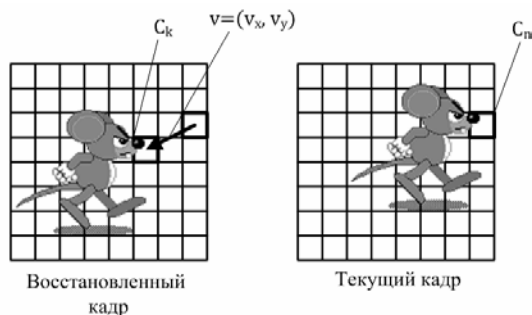


Рис. 1 – Оценка движения базовым блочным методом

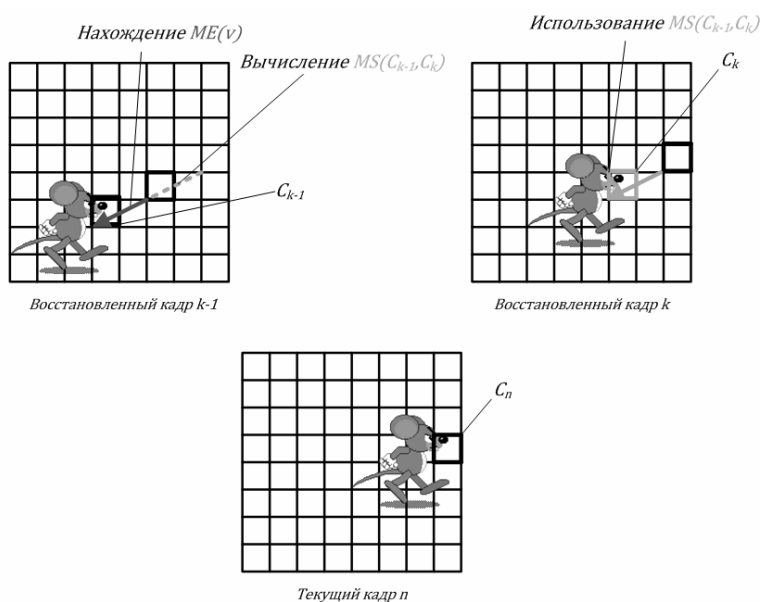


Рис. 2 – Межкадровое движение блока

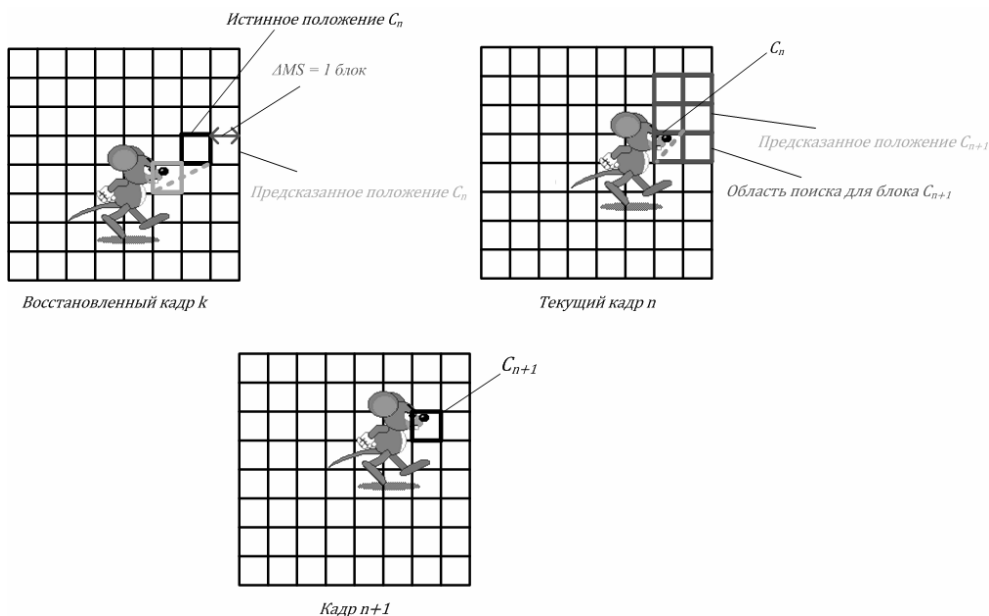


Рис. 3 – Область поиска около предсказанного блока – $\Delta MS(C, C')$

От типа движения объектов в смежных кадрах видеопоследовательности зависит величина отклонения найденного (с помощью функции "скорости" движения) вектора движения от лучшего из возможных векторов движения, т.е. межкадровое изменение "скорости" движения. Обозначим его как $\Delta MS(C, C')$:

$$\Delta MS(C, C') = \sqrt{(C_x - C'_x)^2 + (C_y - C'_y)^2}, \quad (2)$$

где C'_x, C'_y – координаты x и y соответственно предсказанного блока; C_x, C_y – координаты x и y соответственно фактического блока, дающего лучший вектор движения.

Рассмотрим рис. 2, на котором представлено движение объектов изображения первого типа. При статичной камере и движении только объектов изображения величина $\Delta MS(C, C')$ минимальна по сравнению со вторым и третьим типом движения. Движение камеры вносит значительное межкадровое изменение "скорости" движения. Область поиска Ω векторов движения около предсказанного блока ограничим значением $\Delta MS(C, C')$ (рис. 3).

Таким образом, модифицированный блочный метод ОД заключается в том, что для первой пары кадров видеопоследовательности ОД блоков выполняется полным перебором в области фиксированного размера, который покрывает 128 пикселей около соответствующих блоков. Затем для каждого блока этих кадров видеопоследовательности вычисляется значение функции "скорости" движения $MS(C_n, C_k)$, предсказывающее соответствующий вектор движения для следующего кадра. Для каждого блока каждой следующей пары смежных кадров выполняется вычисление межкадрового изменения "скорости" движения $\Delta MS(C, C')$, а затем – поиск вектора движения на основе предсказанного вектора движения в окрестности $\Omega = \Delta MS(C, C')$.

Предложенный модифицированный блочный метод оценки движения, в отличие от базового метода, использует начальное приближение поиска и предсказываемое сокращение области поиска для каждого блока кадра. За счет этого кодеру необходимо выполнить значительно меньшее количество вычислений функции ОД, следовательно, затратить меньше времени на выполнение операции ОД по сравнению с базовым методом.

2.2 Реализация модифицированного блочного метода оценки движения в видеопоследовательностях

Для реализации предложенного блочного метода оценки движения, разработан следующий алгоритм:

Шаг 1 (Инициализация)

Задать N, k, MS .

Инициализировать $\Delta MS = 128$.

Задать массивы $VS_N, F^1, F^2, C_n, C_k, C, C'$.

Шаг 2 (Декодирование первой пары кадров)

$k := 2$.

$F^1 := VS_1$.

$F^2 := VS_2$.

Разбить F^1 на блоки C_k, F^2 – на блоки C_n .

Шаг 3 (Оценка движения первой пары кадров)

Для каждого блока C_n :

Найти блок C_k , дающий лучшее значение $ME(v)$.

Вычислить $MS(C_n, C_k)$.

Кодирование межкадровой разницы.

$F^1 := F^2$.

Шаг 4 (Декодирование очередного кадра)

$k := k + 1$.

Если $k \leq N$, то

$F^2 := VS_k$.

Разбить F^1 на блоки C_k ,

F^2 – на блоки C_n .

Иначе – перейти к шагу 6.

Шаг 5 (Оценка движения)

Для каждого блока C_n :

Найти блок C_k , дающий лучшее значение $ME(v)$ с учетом $MS(C_n, C_k)$ в окрестности $\Delta MS(C, C')$.

Вычислить новое значение $\Delta MS(C, C')$.

Вычислить новое значение $MS(C_n, C_k)$.

Кодирование межкадровой разницы.

$F^1 := F^2$.

Перейти к шагу 4.

Шаг 6 (Завершение алгоритма)

Оценка движения в видеопоследовательности выполнена.

2.3 Параллельный алгоритм оценки движения в видеопоследовательностях

Проанализируем разработанный в подразделе 2.2 алгоритм: выделим те операции, которые можно распараллелить на потоки графического процессора, и те, которые – нельзя. Между кадрами видеопоследовательности существует строгая очередность при кодировании (сжимаемые кадры ссылаются на восстановленные), поэтому декодировать их и разбивать на блоки для последующего сжатия (шаги 2 и 4) нужно последовательно. Операции нахождения для каждого блока векторов движения, вычисления функции "скорости" движения и межкадрового изменения "скорости" движения в шагах 3 и 5 можно выполнять параллельно на потоках графического процессора, поскольку в операциях оценки движения и вычисления значений MS и ΔMS потоки не обмениваются данными между собой. Точка барьерной синхронизации потоков находится перед операцией "Кодирование межкадровой разницы" шагов 3, 5 алгоритма, потому что для выполнения этой операции необходимы все вектора движения кадра. UML-диаграмма деятельности предложенного параллельного алгоритма приведена на рис. 4.

Таким образом, в предложенном алгоритме распараллелены на потоки операции нахождения векторов движения, вычисления функций "скорости" движения блоков и межкадровых изменений "скоростей" движения, т.е. операции алгоритма, обладающие наибольшей вычислительной сложностью. Следует отметить важность выбора функции сравнения векторов движения, оказывающей влияние, как на качество результата ($PSNR$), так и на быстродействие параллельной реализации.

3. Экспериментальная проверка предложенного метода и разработанного алгоритма

Для реализации предложенного алгоритма на графическом процессоре использована аппаратно-программная технология *NVIDIA CUDA* [10]. Основные достоинства этой технологии:

1. Унифицированная технология *CUDA* основана на расширении языка Си, даёт возможность организации доступа к набору инструкций *GPU* и управления его памятью при организации параллельных вычислений;

2. Оптимизированный обмен данными между центральным процессором (ЦП или *CPU*) и *GPU*;

3. Поддержка 32- и 64-битных операционных систем: *Windows*, *Linux* и *Mac OS*;

4. Отсутствие необходимости в графических *API* с дополнительной избыточностью и накладными расходами.

В экспериментах использован компьютер следующей конфигурации:

- центральный процессор *Intel E6420*;
- оперативная память 2 Гб *DDR-2*;
- графический процессор *NVIDIA GTX285* (240 мультипроцессоров).

Видеопоследовательности для экспериментов получены после разделения общепринятых тестовых видеопоследовательностей предложенным в [7] методом.

Настройки кодера, реализующего базовый и предложенный методы ОД, следующие:

- в группе кадров только "P"-кадры;
- целопиксельная точность для ОД полным перебором;
- блоки размером 8×8 пикселей;
- максимальная $\Omega = 32$ пиксела.

В основе всех блочных методов ОД лежит сравнение вычисленных векторов движения по определенному критерию. Критерий сравнения векторов движения – это минимальное значение функции ОД. В качестве функции ОД можно использовать следующие наиболее распространенные функции [11]:

1. Сумма абсолютных разностей значений пикселей блоков (*SAD*):

$$SAD(v) = \sum_{(x,y) \in C_n} |C_n(x,y) - C_k(x+v_x, y+v_y)|, \quad (3)$$

где $v = (v_x, v_y)$ – вектор движения; $C_n(x, y)$ – значения яркостных и цветоразностных компонент пиксела с координатами (x, y) в блоке C_n ; $C_k(x+v_x, y+v_y)$ – значения яркостных и цветоразностных компонент пиксела с координатами $(x+v_x, y+v_y)$ в блоке C_k , смещенном на вектор движения v .

Как видно из формулы (3), для нахождения *SAD* между блоками нужны только операции вычисления модуля, сложения и вычитания. Относительно низкая вычислительная сложность *SAD* обуславливает ее широкую распространенность [12];

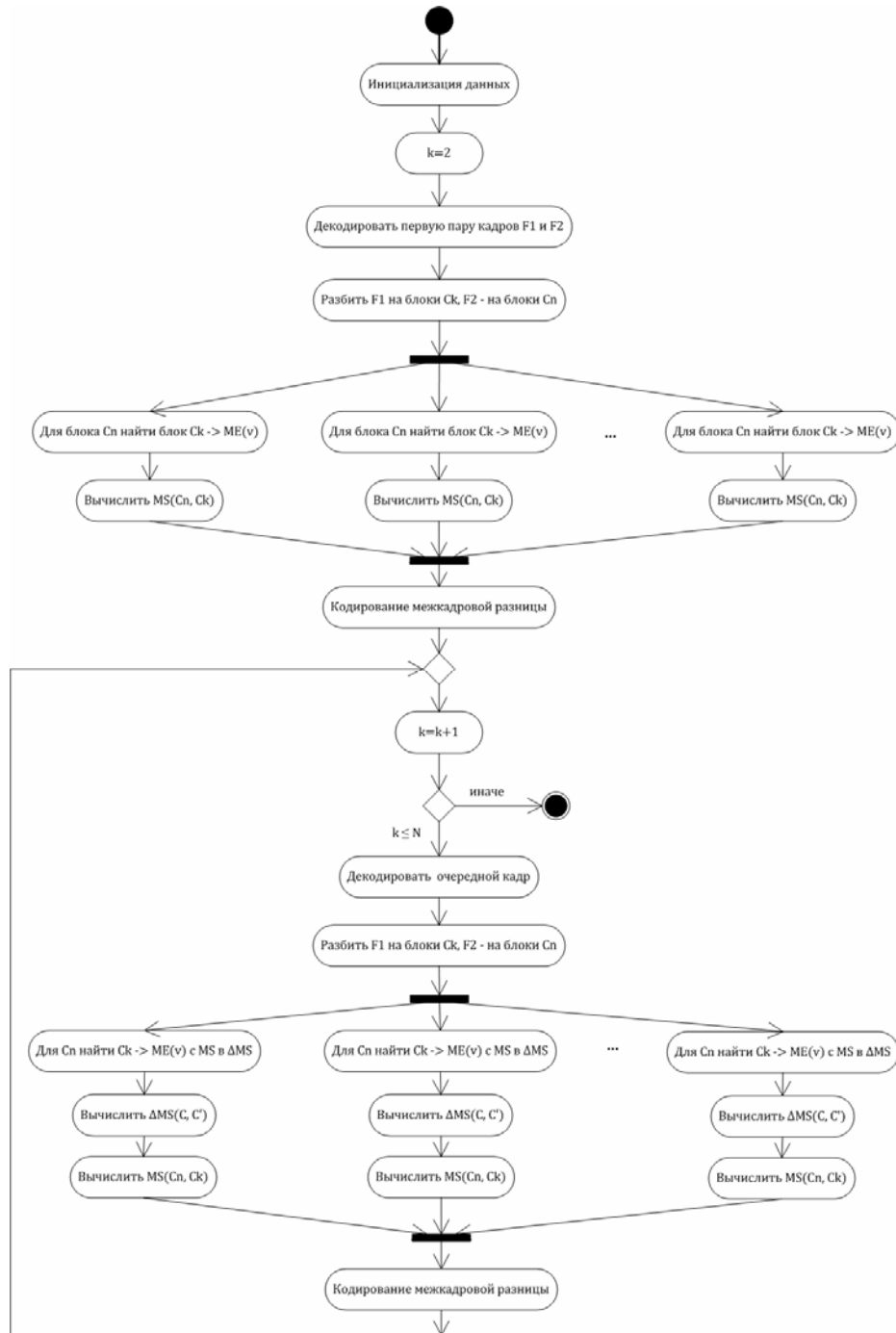


Рис. 4 – UML-диаграмма деятельности предложенного параллельного алгоритма

2. Суммарная квадратичная ошибка значений пикселей блоков (SSE):

$$SSE(v) = \sum_{(x,y) \in C_n} [C_n(x,y) - C_k(x+v_x, y+v_y)]^2. \quad (4)$$

В отличие от SAD , для вычисления SSE требуется выполнение операции умножения вместо взятия модуля;

3. Среднеквадратичное отклонение (MSE):

$$MSE(v) = \frac{1}{S_C} \cdot \sum_{(x,y) \in C_n} [C_n(x,y) - C_k(x+v_x, y+v_y)]^2, \quad (5)$$

где S_C – площадь блока кадра.

В отличие от SSE , для вычисления MSE требуется так же выполнение операции деления.

Для анализа влияния функции сравнения векторов движения на качество сжатия и на

время сжатия видеопоследовательностей по предложенному блочному методу ОД, реализованному на *GPU* с помощью приведенного выше алгоритма, выполнен эксперимент, результаты которого представлены в таблице 1.

Результаты эксперимента позволяют сделать вывод о том, что по критерию качество сжатия-быстродействие, наиболее эффективной функцией сравнения векторов движения для предложенного метода блочной ОД является функция *SAD*. Для последующих экспериментов использована эта функция.

Сравнение предложенного метода ОД с базовым методом по качеству сжатия и по времени сжатия видеопоследовательностей выполнено экспериментально на *GPU*. Непосредственно поиск векторов движения в обоих методах реализован с помощью алгоритма полного перебора в области поиска Ω . При этом базовый метод предполагает фиксированную величину Ω , а модифицированный метод – динамически изменяемое значение Ω для каждого блока. Результаты эксперимента приведены в таблице 2.

Результаты эксперимента позволяют сделать вывод о том, что предложенный метод сокращает время сжатия тестовой видеопоследовательности на *GPU* относительно базового метода в среднем в 10,33 раза для стандарта *SDTV* и в среднем в 9,8 раза для *HDTV*. При этом предложенный метод ведет к деградации качества сжатия видеопоследовательностей, которая в среднем составляет 7,53% относительно

качества видеопоследовательностей, сжатых с применением базового метода ОД.

Чем интенсивнее движение в видеопоследовательности, тем значительнее деградация качества. Следовательно, предложенный метод наиболее эффективен для видеопоследовательностей, не содержащих интенсивное движение.

Экспериментально установлено, что чем интенсивнее движение блоков в смежных кадрах, тем больше межкадровое изменение "скорости" движения ΔMS и тем шире область поиска, соответственно, тем больше время сжатия и меньше выигрыш от применения предложенного метода.

Сравнение суммарного времени сжатия всех задействованных в экспериментах тестовых видеопоследовательностей на *CPU* и на *GPU* с помощью базового блочного метода ОД и предложенного метода выполнено экспериментально. Для *CPU* векторные расширения не использовались. Результаты эксперимента в логарифмической шкале приведены на рис. 5.

Таким образом, применение предложенного модифицированного блочного метода ОД сократило суммарное время сжатия тестовых видеопоследовательностей на *CPU* в 10,63 раза, а на *GPU* – в 9,85 раза. Так же экспериментально установлено, что благодаря предложенному методу, *GPU* дал ускорение в 250 раз относительно *CPU* процесса сжатия тестовых видеопоследовательностей.

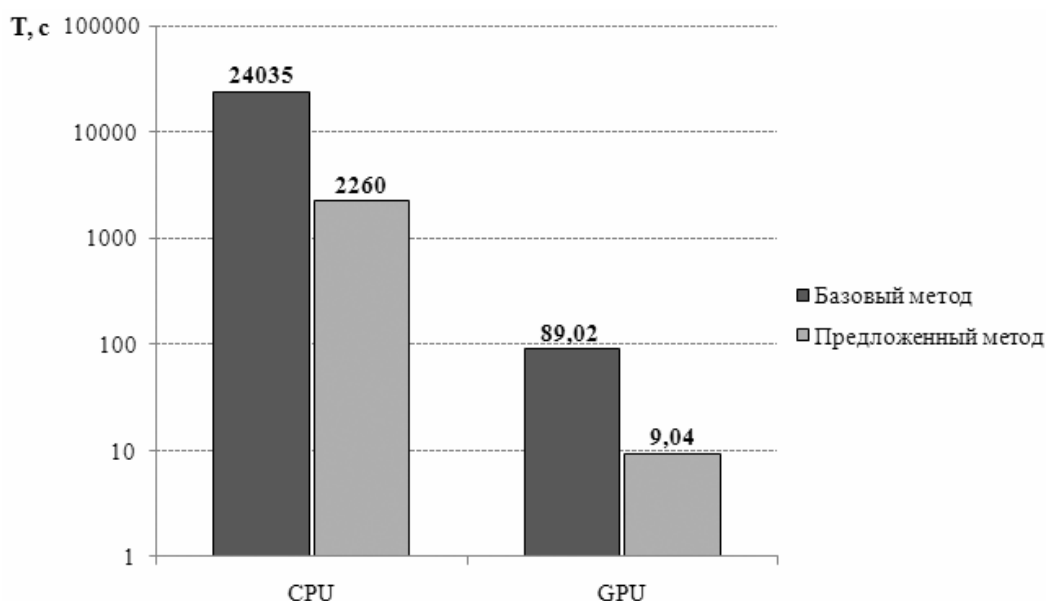


Рис. 5 – Суммарное время сжатия всех тестовых видеопоследовательностей на *CPU* и на *GPU*

Таблица 1 – Влияние функции сравнения векторов движения на качество и на время сжатия видеопоследовательностей по предложенному методу ОД

Тестовая видеопоследовательность	Стандарт представления	Качество сжатия, <i>PSNR</i> дБ			Время сжатия на <i>GPU</i> , с		
		<i>SAD</i>	<i>SSE</i>	<i>MSE</i>	<i>SAD</i>	<i>SSE</i>	<i>MSE</i>
<i>Akiyo Carphone Claire part1</i>	<i>SDTV</i>	32,8	33,2	33,2	0,04	0,03	0,04
<i>Akiyo Carphone Claire part2</i>	<i>SDTV</i>	26,85	27,09	27,09	0,03	0,03	0,03
<i>Akiyo Carphone Claire part3</i>	<i>SDTV</i>	31,59	32	32	0,07	0,07	0,08
<i>Coastguard Waterfall Container part1</i>	<i>SDTV</i>	24,41	24,47	24,47	0,04	0,04	0,04
<i>Coastguard Waterfall Container part2</i>	<i>SDTV</i>	29,6	29,69	29,69	0,04	0,05	0,05
<i>Coastguard Waterfall Container part3</i>	<i>SDTV</i>	31,65	31,79	31,79	0,05	0,05	0,06
<i>Silent Mother and Daughter Bus part1</i>	<i>SDTV</i>	31,82	31,91	31,91	0,05	0,05	0,05
<i>Silent Mother and Daughter Bus part2</i>	<i>SDTV</i>	36,55	36,7	36,7	0,06	0,05	0,06
<i>Silent Mother and Daughter Bus part3</i>	<i>SDTV</i>	20,44	20,41	20,41	0,04	0,04	0,05
<i>Mobile Bridge Paris part1</i>	<i>SDTV</i>	20,5	20,6	20,64	0,04	0,04	0,04
<i>Mobile Bridge Paris part2</i>	<i>SDTV</i>	30,62	30,77	30,78	0,17	0,18	0,18
<i>Mobile Bridge Paris part3</i>	<i>SDTV</i>	25,89	26,05	26,07	0,05	0,05	0,05
<i>News Tempete Hall monitor part1</i>	<i>SDTV</i>	30,83	31,08	31,08	0,07	0,07	0,08
<i>News Tempete Hall monitor part2</i>	<i>SDTV</i>	29,69	29,81	29,81	0,03	0,04	0,04
<i>News Tempete Hall monitor part3</i>	<i>SDTV</i>	23,27	23,43	23,44	0,06	0,06	0,06
<i>News Tempete Hall monitor part4</i>	<i>SDTV</i>	32,18	32,21	32,21	0,05	0,05	0,05
<i>Blue sky Pedestrian area part1</i>	<i>HDTV</i>	26,73	26,85	26,85	1,09	1,15	1,17
<i>Blue sky Pedestrian area part2</i>	<i>HDTV</i>	25,62	25,68	25,68	1,67	1,79	1,82
<i>Riverbed Station part1</i>	<i>HDTV</i>	27,16	27,19	27,2	1,14	1,21	1,23
<i>Riverbed Station part2</i>	<i>HDTV</i>	26,3	26,32	26,32	1,59	1,69	1,72
<i>Rush hour Tractor part1</i>	<i>HDTV</i>	26,1	26,31	26,32	1,06	1,13	1,15
<i>Rush hour Tractor part2</i>	<i>HDTV</i>	25,41	25,47	25,48	1,63	1,74	1,76

Таблица 2 – Сравнение предложенного метода с базовым методом по качеству и времени сжатия

Тестовая видеопоследовательность	Качество сжатия, <i>PSNR</i> дБ		Время сжатия на <i>GPU</i> , с	
	Базовый метод	Предложенный метод	Базовый метод	Предложенный метод
<i>Akiyo Carphone Claire part1</i>	33,11	32,8	0,36	0,04
<i>Akiyo Carphone Claire part2</i>	29,17	26,85	0,46	0,03
<i>Akiyo Carphone Claire part3</i>	32,18	31,59	0,59	0,07
<i>Coastguard Waterfall Container part1</i>	26,64	24,41	0,59	0,04
<i>Coastguard Waterfall Container part2</i>	30,15	29,6	0,53	0,04
<i>Coastguard Waterfall Container part3</i>	31,82	31,65	0,64	0,05
<i>Silent Mother and Daughter Bus part1</i>	34,08	31,82	0,53	0,05
<i>Silent Mother and Daughter Bus part2</i>	37,68	36,55	0,59	0,06
<i>Silent Mother and Daughter Bus part3</i>	23,45	20,44	0,53	0,04
<i>Mobile Bridge Paris part1</i>	21,61	20,5	0,53	0,04
<i>Mobile Bridge Paris part2</i>	31,04	30,62	0,80	0,17
<i>Mobile Bridge Paris part3</i>	27,73	25,89	0,53	0,05
<i>News Tempete Hall monitor part1</i>	32,5	30,83	0,75	0,07
<i>News Tempete Hall monitor part2</i>	33,02	29,69	0,26	0,03
<i>News Tempete Hall monitor part3</i>	24,32	23,27	0,80	0,06
<i>News Tempete Hall monitor part4</i>	34,9	32,18	0,53	0,05
<i>Blue sky Pedestrian area part1</i>	31,42	26,73	12,03	1,09
<i>Blue sky Pedestrian area part2</i>	28,63	25,62	14,74	1,67
<i>Riverbed Station part1</i>	32,1	27,16	12,37	1,14
<i>Riverbed Station part2</i>	30,8	26,3	14,55	1,59
<i>Rush hour Tractor part1</i>	31,12	26,1	11,79	1,06
<i>Rush hour Tractor part2</i>	28,16	25,41	14,48	1,63

4. Анализ накладных расходов параллельной системы на базе GPU

Выполним теоретическую оценку максимально возможного ускорения предложенного алгоритма по закону Амдала (будем считать, что все операции равноценны по вычислительной сложности). Введем следующие обозначения: S_p – теоретическое ускорение, которое может быть получено на p – потоках параллельной системы; α – доля последовательных операций алгоритма, а $(1-\alpha)$ – доля параллельных операций алгоритма.

Количество потоков GPU, выполняющих предложенный алгоритм, возьмем равным количеству блоков, на которые разбивается кадр видеопоследовательности. Например, для кадра видеопоследовательности стандарта высокой четкости размером 1920×1080 пикселей количество блоков размером 8×8 пикселей составляет ровно 32400. Оценим долю последовательных и параллельных операций алгоритма. Количество параллельных операций, выполняемых потоками графического процессора:

$$N_p = N \cdot (66049 \cdot ME + 6) + (N_F - 2) \times \\ \times N \cdot [(2 \cdot \Delta MS + 1)^2 \cdot ME + 12], \quad (6)$$

где N_p – количество параллельных операций алгоритма; N – количество блоков кадра; ME – количество операций функции ОД; N_F – количество кадров видеопоследовательности; ΔMS – межкадровое изменение "скорости" движения.

В предложенном алгоритме оценка движения первой пары кадров выполняется с помощью поиска в области фиксированного размера: для каждого блока выполняется 66049 операций вычисления функции ОД, т.е. для области поиска, покрывающей 128 пикселей $(2 \cdot 128 + 1)^2 = 66049$ и шесть операций вычисления функции "скорости" движения блоков. Для остальных кадров, которых осталось $(N_F - 2)$ штук, для каждого блока выполняется $(2 \cdot \Delta MS + 1)^2$ операций вычисления функции ОД, шесть операций вычисления функции "скорости" движения блоков и шесть операций вычисления изменения "скорости" движения.

Количество последовательных операций алгоритма:

$$N_S = 5 + (N_F - 1) \cdot 5 = 5 \cdot N_F, \quad (7)$$

где N_S – количество последовательных операций алгоритма.

В количество последовательных операций входят первые пять операций: инициализация данных, декодирование первой пары кадров, разбиение кадров на блоки, кодирование межкадровой разницы. Следующие пять операций выполняются $(N_F - 1)$ раз: инкремент счетчика кадров k , сравнение k и N_F , декодирование следующего кадра, разбиение кадров на блоки, кодирование межкадровой разницы.

Доля последовательных операций алгоритма:

$$\alpha = \frac{N_S}{N_S + N_P} = \quad (8)$$

$$= \frac{5 \cdot N_F}{5 \cdot N_F + N \cdot [66049 \cdot ME + 6 + (N_F - 2) \cdot ((2 \cdot \Delta MS + 1)^2 \cdot ME + 12)]}.$$

Так, для функции ОД – SAD, количество операций $ME = 3 \cdot S_C = 3 \cdot 64 = 192$. Для медленного движения блоков в смежных кадрах $\Delta MS = 8$. Тогда получим оценку α , зависящую от N и N_F :

$$\alpha = \quad (9)$$

$$= \frac{5 \cdot N_F}{5 \cdot N_F + N \cdot [12,68 \cdot 10^6 + 55,5 \cdot 10^3 \cdot (N_F - 2)]}.$$

Подставив $N = 32400$ блоков, $N_F = 55$ кадров тестовой видеопоследовательности [7], получим $\alpha \approx 5,43 \cdot 10^{-10}$, т.е. доля последовательных операций ничтожно мала (с допущением того, что все операции равноценны). Следовательно, доля параллельных операций $(1 - \alpha) \approx 1$ и $S_p \approx p$. Эффективность параллельной системы

$E_p = \frac{S_p}{p}$ при $S_p \approx p$ будет примерно равна единице, т.е. $E_p \approx 1$.

Оценка эффективности предложенного алгоритма показывает его теоретическую способность продемонстрировать значительное ускорение на графическом процессоре.

Накладные расходы, снижающие ускорение и эффективность применения GPU, включают затраты времени на пересылки данных из оперативной памяти компьютера в память GPU, доступ потоков к памяти, синхронизацию потоков. Проанализируем накладные расходы применения распределенной системы [13]. Пусть на вычислительный узел распределенной сис-

темы (рис. 6) для обработки поступают данные (видеоинформация для сжатия), и затем результаты обработки в виде данных пересылаются другому узлу. Тогда максимальное суммарное время T_{CN} выполнения коммуникационной составляющей такой операции определим по следующей формуле:

$$T_{CN} = 2(T_1 + T_2 + T_3 + T_4 + T_5 + T_6). \quad (10)$$

Если опциональные компоненты вычислительного узла отсутствуют, то T_2 и T_6 в (10) не учитываются. При интенсивном обмене данными доля T_1 , T_2 и T_3 в суммарном времени T_{CN} значительна. Возможны случаи, когда T_{CN} может превышать время выполнения целевых вычислений на *CPU* и *GPU* [14]. Величины $T_1 \dots T_6$ в современных материнских платах с медными проводниками составляют единицы микросекунд, что также может служить причиной снижения общей производительности системы.

Рассмотрим подробнее барьерную синхронизацию потоков *GPU*, выполняемую перед каждым кодированием межкадровой разницы. Каждый поток обрабатывает один блок кадра – находит лучший по функции ОД вектор движения, вычисляет MS и ΔMS . Поскольку для разных блоков время нахождения лучшего вектора движения разное, то некоторым потокам потребуется меньше времени для выполнения этих операций, а некоторым – больше. Для кодирования межкадровой разницы необходимы все вектора движения блоков кадра, поэтому в этом месте предложенного алгоритма устанавливается точка барьерной синхронизации потоков, обуславливающая накладные расходы на синхронизацию потоков.

Таким образом, доля накладных расходов в параллельной системе на базе *GPU*, обусловленных пересылками данных между ОЗУ и *GPU*, а так же синхронизацией потоков *GPU*, может быть существенной по отношению к доле целевого вычисления. Несмотря на это, производительность *GPU*, который реализует представленный метод, значительно выше производительности *CPU*.

Применение *GPU* на узлах распределенной системы для сжатия видеопоследовательностей по предложенному методу позволяет сжимать видеопоследовательности с такой производительностью, которая обеспечивает появление сжатых кадров со скоростью,

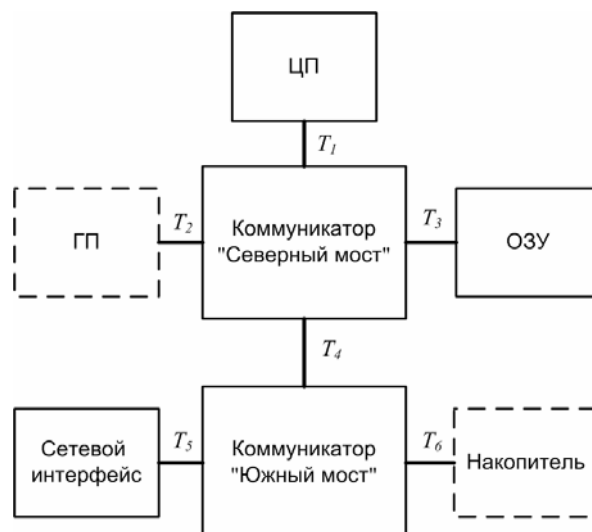


Рис. 6 – Структура типового чипсета материнской платы вычислительного узла

не отстающей от частоты смены кадров видеопоследовательности, т.е. в реальном времени без существенной потери качества результата.

Однако, применение *GPU* на узлах распределенной системы для сжатия видеоинформации по предложенному методу целесообразно за исключением видеопоследовательностей, не допускающих деградацию качества.

Результаты и выводы

В ходе исследования были получены следующие результаты:

1. Разработан модифицированный блочный метод ОД, в основе которого лежит вычисление для каждого блока кадра видеопоследовательности значения функции "скорости" движения и межкадрового изменения "скорости" движения, ограничивающего область поиска;
2. Данный метод ОД применим для видеопоследовательностей, предварительно разделенных на части в распределенной системе при помощи предложенного в [7] метода;
3. В методе используется начальное приближение поиска и предсказываемое сокращение области поиска для каждого блока кадра. За счет этого кодеру необходимо выполнить значительно меньшее количество вычислений функции ОД, следовательно, – затратить меньше времени на выполнение ОД по сравнению с базовым методом;
4. Разработан параллельный алгоритм, реализующий предложенный метод, на *GPU*. Ис-

следовано влияние функции сравнения векторов движения в предложенном методе на время сжатия и качество сжатия. Наиболее эффективной по критерию качество сжатия к быстрдействию функцией сравнения векторов движения для предложенного метода является функция *SAD*;

5. Экспериментально установлено, что предложенный метод сократил время сжатия тестовой видеопоследовательности на *GPU* относительно базового метода в среднем в 10,33 раза для стандарта *SDTV* и в среднем в 9,8 раза для *HDTV*. При этом метод привел к деградации качества сжатия видеопоследовательностей, которая в среднем составила 7,53% относительно качества видеопоследовательностей, сжатых с применением базового метода ОД;
6. Экспериментально установлено, что предложенный метод блочной ОД обеспечивает возможность сжатия видеoinформации стандарта *HDTV* в реальном времени на *GPU* с приемлемой деградацией качества.

СПИСОК ЛИТЕРАТУРЫ

1. Generic coding of moving pictures and associated audio information : ISO/IEC 13818 (MPEG-2), 1994.
2. Advanced video coding for generic audiovisual services : ITU-T Recommendation H.264 and ISO/IEC 14496-10 (AVC), 2003.
3. Bhaskaran V. Image and video compression standards: algorithms and architectures / V. Bhaskaran, K. Konstantinides. – MA, USA : Kluwer Academic Publishers Norwell, 1997. – 454 p.
4. Гришин С.В. Обзор блочных методов оценки движения в цифровых видео сигналах / С.В. Гришин, Д.С. Ватолин, А.С. Лукин, С.Ю. Путилин, К.Н. Стрельников // Программные системы и инструменты. Тематический сборник №9. – 2008. – С. 50–62.
5. Performance analysis and architecture evaluation of MPEG-4 video codec system : IEEE International Symposium on Circuits and Systems / H.-C. Chang,

L.-G. Chen, M.-Y. Hsu, Y.-C. Chang. – Geneva, 2000. – P. 449–452.

6. Parameter values for the HDTV standards for production and international programme exchange : ITU-R Recommendation BT.709, 2008.
7. Скрупский С.Ю. Повышение эффективности сжатия видеoinформации в распределенных компьютерных системах / С.Ю. Скрупский // Электронное моделирование. – 2011, №6 (33). – С. 57–72.
8. Скрупский С.Ю. Параметры компрессии видеoinформации в распределенных системах / С.Ю. Скрупский, Н.В. Луценко, Л.С. Скрупская // Радиоелектроника, информатика, управління. – 2011. – №2 (25). – С. 137–143.
9. Рабинович А.В. Алгоритмы оценки движения для систем сжатия ТВ изображений / А.В. Рабинович // Информсвязь № 2241. – 2004. – С. 26–34.
10. NVIDIA Corporation. NVIDIA CUDA Compute Unified Device Architecture Programming Guide. Ver 2.3. / NVIDIA Corporation. : NVIDIA, 2009. – 111 p.
11. Рабинович А.В. Критерии оценки движения для алгоритмов сжатия изображений / А.В. Рабинович // Информсвязь № 2241. – 2004. – С. 35–41.
12. Complexity and PSNR-Comparison of several Fast Motion Estimation Algorithms for MPEG-4 : Proc. SPIE "Applications of Digital Image Processing XXI", San Diego CA, 21–24 July 1998 / Peter M. Kuhn, Georg Diebel, Stephan Herrmann, Andreas Keil, Hubert Mooshofer, André Kaup, Robert Mayer, Walter Stechele. – San Diego CA, 1998. – Vol. 3460, P. 486–499.
13. О влиянии среды передачи данных на производительность вычислительных систем : Материалы четвертой Всеукраинской научно-практической конференции молодых ученых и студентов "Информационные процессы и технологии "Информатика – 2011", 25–29 апреля 2011 г. / С.Ю. Скрупский, С.С. Грушко, О.В. Польская. – Севастополь, 2011. – С. 77–79.
14. Скрупский С.Ю. SIMD–реализация алгоритма SAXPY на GPU / С.Ю. Скрупский, Р.К. Кудерметов // ВОСТОЧНО-ЕВРОПЕЙСКИЙ ЖУРНАЛ ПЕРЕДОВЫХ ТЕХНОЛОГИЙ. – 2009, №40. – С. 42–46.

S.Y. Skrupsky. **Improving the performance of videoinformation compression distributed systems.**

С.Ю. Скрупський. **Підвищення продуктивності розподілених систем компресії відеоінформації.**

Разработан модифицированный блочный метод оценки движения в видеопоследовательностях. Выполнено распараллеливание алгоритма, реализующего разработанный метод на графическом процессоре. Экспериментально исследованы преимущества и недостатки представленных метода и алгоритма. Предложенный метод позволяет выполнять компрессию видеoinформации стандарта высокой четкости в реальном времени без значительной потери качества результата.