УДК 004.451.35+004.932:532.574

Мочалин А.Е.,
д.т.н. Мочалин Е.В.
(ДонГТУ, г. Алчевск, Украина)

ИСПОЛЬЗОВАНИЕ МЕХАНИЗМОВ УПРАВЛЕНИЯ ВИРТУАЛЬНОЙ ПАМЯТЬЮ В WIN32 ДЛЯ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ЦИФРОВЫХ ТРАССЕРНЫХ ИЗОБРАЖЕНИЙ

Показана можливість суттєвого зниження обчислювальних витрат на обробку цифрових трасерних зображень за новим методом, що забезпечує більшу точність та менші вимоги до реєструючої апаратури. Результат досягається за рахунок оптимального керування віртуальною пам'яттю у сполученні зі структурною обробкою виключень в середовищі WIN32.

Ключові слова: *віртуальна пам'ять, обробка виключень, операційна система, трасерне зображення.*

Показана возможность существенного снижения вычислительных затрат на обработку цифровых трассерных изображений по новому методу, обеспечивающему большую точность и меньшие требования к регистрирующей аппаратуре. Результат достигается за счет оптимального управления виртуальной памятью в сочетании со структурной обработкой исключений в среде WIN32.

Ключевые слова: *виртуальная память, обработка исключений, операционная система, трассерное изображение.*

Введение. При выполнении фундаментальных и прикладных научных исследований, в процессе разработки новой техники и создания новых технологий большое значение имеет экспериментальное определение полей скорости текучих сред в различных областях. Одним из наиболее перспективных и интенсивно развивающихся подходов в этой области является цифровая трассерная визуализация, на которой, в частности, основан метод, получивший международное название Particle image velocimetry (PIV) [1,2]. Составной частью этого метода является компьютерная обработка цифровых изображений, получаемых фотографированием подсвеченного ярким световым лучом сечения измеряемого потока с подмешанными в него мелкими частицами – трассерами. Рассматриваются два кадра, соответствующие двум близким моментам времени. Базовым или стандартным в настоящее время принято считать

алгоритм обработки, основанный на разбиении всей области изображения на элементарные расчетные окна и вычислении кросс-корелляционной функции для каждого окна на первом кадре и соответствующего ему окна на втором кадре по формуле [1]:

$$R(m,n) = \frac{\sum_{i=0}^{M}\sum_{j=0}^{N} f_1(i,j) f_2(i+m, j+n)}{\sum_{i=0}^{M}\sum_{j=0}^{N} f_1(i,j) f_2(i,j)}, \qquad (1)$$

где $f_1(i,j), f_2(i,j)$ − значения интенсивности пикселя с локальными координатами (i,j) на первом и втором кадрах соответственно; m,n − горизонтальное и вертикальное целочисленные смещения пары окон друг относительно друга.

Значения m,n, соответствующие максимуму функции (1), определяют целочисленное смещение элементарного окна разбиения, по которому вычисляется вектор скорости в точке, совпадающей с центром окна. Путем интерполяции значений кросс-корреляционной функции в соседних точках определяется дробная часть смещения в долях пикселя. Схема алгоритма, заимствованная из [3], представлена на рисунке 1.

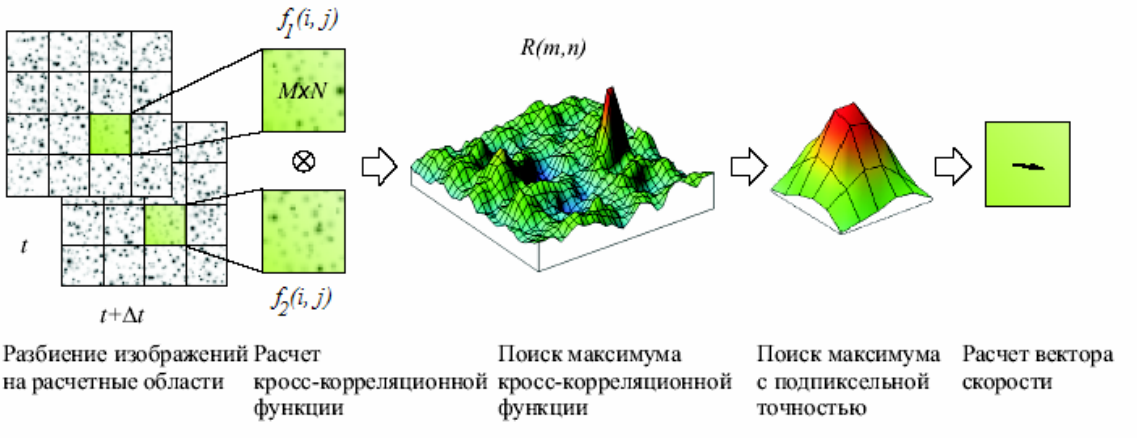

Рисунок 1 − Схема стандартного кросс-корреляционного алгоритма определения векторов скорости по изображениям трассеров [3]

Одним из главных недостатков стандартного алгоритма, как показано в работах [4,5], является низкая точность при анализе потоков с

большими локальными градиентами скорости. В этом случае область, представленная на первом кадре прямоугольным элементарным окном, заметно деформируется и на втором кадре уже не представима с достаточной точностью прямоугольным окном той же формы, смещенным относительно исходного положения на определенную величину. В то же время стандартный алгоритм не учитывает деформации элементарных окон. В полной мере эта проблема еще не нашла общепринятого решения.

В [4,5] предложен итерационный алгоритм, предусматривающий построение реконструированного кадра на основе первого из двух последовательных снимков и текущего приближения поля скоростей. Уточнение значений скорости осуществляется путем сравнения реконструированного кадра и второго из имеющихся изображений. Этот метод, среди прочих преимуществ, учитывает наличие градиентов скорости во всей области, что отражается при построении реконструированного кадра. Однако компьютерная реализация нового метода связана с существенно большими требованиями к вычислительным ресурсам, чем реализация стандартного кросс-корреляционного алгоритма. Так, в работе [6], посвященной описанию близкого по идеологии алгоритма, также учитывающего деформацию исходного шаблона, приводятся данные о том, что затраты машинного времени на обработку одной пары снимков, при использовании APOLLO – WORKSTATION DN 10000 (с производительностью в 20 раз большей, чем у персональных компьютеров в 1993 г.), составили 12 суток. Даже с учетом прогресса компьютерной техники за 17 лет, использование только лишь новой аппаратной платформы не позволит свести затраты до такого уровня, который необходим для реализации PIV – исследований турбулентных течений, где необходима обработка большого числа последовательных пар снимков.

Следовательно, прогресс, основанный на реализации нового метода, может быть достигнут на основе оптимизации алгоритма его реализации в совокупности с использованием средств системной поддержки распределения ресурсов, предоставляемых пользователям разработчиками современных операционных систем.

Целью настоящей работы является оптимизация алгоритма обработки цифровых трассерных изображений по методу построения реконструированного кадра с применением развитых способов управления виртуальной памятью в среде WIN32.

Основные результаты. Согласно [4,5], Реконструированное изображение строится на основе первого из двух последовательных кадров путем определения смещения каждого пикселя трассерного изображения по следующим формулам:

$$dx_{ij}^k = X_1^k(1-\xi_1)(1-\xi_2) + X_2^k\xi_1(1-\xi_2) + X_3^k\xi_1\xi_2 + X_4^k\xi_2(1-\xi_1), \quad (2)$$

$$dy_{ij}^k = Y_1^k(1-\xi_1)(1-\xi_2) + Y_2^k\xi_1(1-\xi_2) + Y_3^k\xi_1\xi_2 + Y_4^k\xi_2(1-\xi_1), \quad (3)$$

где dx_{ij}^k, dy_{ij}^k - соответственно горизонтальное и вертикальное перемещение пикселя с координатами i, j, принадлежащего элементарному окну разбиения с номером k, между первым и вторым кадрами; $X_1^k, X_2^k, X_3^k, X_4^k, Y_1^k, Y_2^k, Y_3^k, Y_4^k$ – значения горизонтальных и вертикальных проекций векторов перемещения в угловых точках окна k (рисунок 2).

Нормированные локальные координаты пикселя, в соответствии со схемой, приведенной на рисунке 2, определяются следующим образом:

$$\xi_1 = \frac{i}{M},\ \xi_2 = \frac{j}{N}. \quad (4)$$

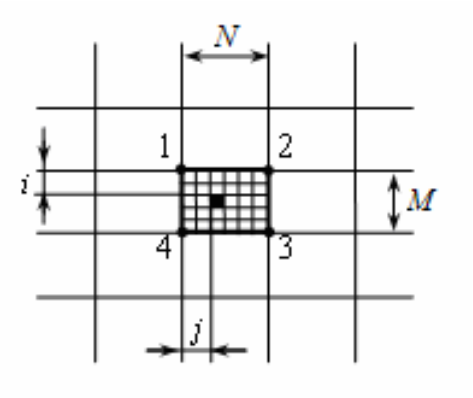

Рисунок 2 − Элементарное окно разбиения на первом кадре

Поскольку перемещения пикселей, определяемые выражениями (2) − (4), являются дробными, то интенсивность каждого смещенного пикселя распределяется по четырем смежным пикселям реконструированного кадра. В итоге, для определения интенсивности пикселя (p,q) реконструированного кадра принято

$$f_r(p,q) = \sum_{k=1}^{K}\sum_{i=1}^{M}\sum_{j=1}^{N} \frac{f_1^k(i,j)(S_{pq})_{ij}^k}{\sum_{k=1}^{K}\sum_{i=1}^{M}\sum_{j=1}^{N}(S_{pq})_{ij}^k}, \quad (5)$$

где $(S_{pq})_{ij}^{k}$ − площадь пересечения пикселя первого кадра с координатами i, j с пикселем p, q на реконструированном кадре, K − количество элементарных окон разбиения рассматриваемой области. Не будем здесь подробно останавливаться на вычислении площадей $(S_{pq})_{ij}^{k}$, необходимые формулы приведены в работе [5].

В соответствии с рассматриваемым подходом уточнение значений перемещений X_k, Y_k для всех узловых точек k (общим числом $K1$) осуществляется путем минимизации функции ошибки методом градиентного спуска.

$$E(X_1, X_2, ...X_{K1}, Y_1, Y_2, ...Y_{K1}) = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(f_2(i,j) - f_r(i,j))^2}{\left(\max_{i,j}(f_2(i,j) - f_r(i,j))\right)^2}} \qquad (6)$$

При этом на каждом шаге итерации требуется многократный пересчет реконструированного кадра на основе формул (2)…(5) с определением соответствующей ошибки по формуле (6), что необходимо для вычисления проекций градиента функции ошибки. Однако, при каждом таком пересчете изменяется только одно из узловых значений проекций перемещения. В целях сокращения объема вычислений следует не проводить их каждый раз для всей области, а в каждом случае определять зону влияния измененного узлового значения и выполнять локальный пересчет реконструированного кадра только в пределах этой зоны влияния. Это один из основных резервов сокращения времени счета, если не считать распараллеливания вычислений, что является темой отдельного исследования.

Кроме времени вычислений рассматриваемая задача предъявляет большие требования и к объему памяти для хранения данных и промежуточных результатов. При размерах цифрового изображения в несколько мегапикселей, что характерно для современной практики применения метода PIV, без тщательного анализа формы представления промежуточных данных (структуры используемых массивов) легко можно прийти к ситуации нехватки физической памяти, либо к замедлению работы с памятью из-за слишком частого обращения к страничному файлу (файлу подкачки). Однако, отмеченное выше обстоятельство, связанное с тем, что при каждом из огромного числа пересчетов ре-

конструированного кадра изменяется только его небольшая область, позволяет экономить и память. Для этого нужно динамически выделять только такой ее объем, который необходим для сохранения изменений в локальной области реконструированного кадра и для сохранения ее исходного состояния. При этом каждый раз текущая локальная область, претерпевающая изменения, не состоит из одной непрерывной группы пикселей подряд расположенных в единой сквозной нумерации. Поэтому, для того, чтобы в процессе вычислений оценивать требуемые объемы динамически выделяемой памяти и использовать выделенные массивы в процессе расчета, необходимы дополнительные вычисления, связанные с определением размеров массивов для промежуточных результатов и для установления соответствия в нумерации изменяемых пикселей в пределах всего реконструированного кадра и рассматриваемой локальной области. Эти вычислительные затраты заметно сказываются на времени счета.

В такой ситуации можно существенно оптимизировать вычислительный процесс за счет использования развитых механизмов управления виртуальной памятью, реализованных в ядре WIN 32 в виде системной поддержки пользователей.

Программа существенно упрощается, а ее быстродействие повышается, если при каждом из многочисленных пересчетов интенсивности отдельных пикселей реконструированного кадра мы будем идентифицировать каждый из них в единой общей нумерации, относящейся ко всему кадру, например, через индексы i, j. Но тогда, на первый взгляд, нам необходимо каждый раз выделять память для всего массива (обозначим его `F_r`) размером $M1 \times N1$ ($M1, N1$ − размеры кадра по вертикали и горизонтали), в то время, как работа будет происходить только с небольшой частью его элементов. Ситуацию можно изменить, используя то обстоятельство, что в WIN32 реализовано разделение операций резервирования регионов виртуальной памяти и передачи этим регионам (или, что существенно, их частям-блокам) физической памяти в страничном файле. Особую гибкость этот механизм приобретает в сочетании со структурной обработкой исключений (SEH), возможность которой заложена в операционной системе (ОС) и поддерживается на уровне компиляторов (в частности, Microsoft Visual C++).

Рассмотрим наиболее характерную операцию записи нового значения `new_value` интенсивности пикселя (0…256) в элемент массива `F_r[i,j]`. Сначала мы резервируем регион виртуальной памяти достаточный для размещения всего массива `F_r[M1,N1]`

...

```
    PBYTE    pF_r    =(PBYTE)    VirtualAlloc(NULL,
(SIZE_T)M1*N1,
    MEM_RESERVE, PAGE_READWRITE);
    …
```

При этом физическая память региону не передается, а только активируется диапазон виртуального адресного пространства размером $M1 \times N1$, увеличенный для кратности размеру страницы памяти (4 Кб для процессоров Intel).

Обращение к элементу `F_r[i,j]` оформляем следующим образом (приведем пример для случая записи):

```
    …
    __try
    {
       *(pF_r+i*N1+j) = (BYTE) new_value;
    }
__except (EXCEPTION_EXECUTE_HANDLER)
    {
       VirtualAlloc((PVOID)(pF_r+i*N1+j),
4*1024, MEM_COMMIT, PAGE_READWRITE);
    *(pF_r+i*N1+j) = (BYTE) new_value;
    }
    …
```

Синтаксис структурной обработки исключений основан на использовании блоков *try−except*, поддерживаемых Windows совместимыми C++ компиляторами. В приведенной выше конструкции первое обращение к элементу массива осуществляется в блоке *try* . В том случае, если физическая память по указанному адресу уже передана ранее, присвоение выполнится успешно и блок *except* не будет выполняться. Если же физическая память по этому адресу еще не выделена, выполнение этой команды вызовет процессорное исключение и система, при значении фильтра `EXCEPTION_EXECUTE_HANDLER` (-1), передаст управление командам, расположенным в блоке *except* и представляющих собой обработчик исключения. Обработка исключения заключает-

ся в системном вызове `VirtualAlloc`, который при указанном значении аргументов приведет к выделению физической памяти блоку минимального размера (одна страница), содержащему необходимый адрес. При этом, начальный адрес блока будет равен ближайшему меньшему значению, кратному размеру страницы. После передачи блоку физической памяти реализуется повторное обращение по требуемому адресу, которое уже будет успешным.

Хотя выделенный блок памяти имеет размер существенно больший необходимого для размещения 1-го элемента массива (в нашем случае 1 б), однако в этом есть определенное преимущество. Вероятность того, что в ближайшее время будет осуществляться обращение по адресам, близким к рассматриваемому в данный момент адресу, весьма велика. И в этом случае первое обращение к памяти не вызовет исключения и частота вызова функции `VirtualAlloc` существенно снизится, что ускоряет работу программы. Более детально затронутые вопросы управления виртуальной памятью и реализации SHE рассмотрены в работе [7].

Выводы. Рассмотренный способ динамического использования памяти в Win32 позволяет эффективно организовать работу с данными, организованными в виде большого количества массивов большой размерности в ситуации, когда в большинстве случаев приходится работать с небольшими участками этих массивов. Резервирование регионов виртуального адресного пространства под целые массивы снимает проблему обеспечения соответствия элементов массивов различной структуры и размерности. Использование структурной обработки исключений оптимизирует адресное выделение физической памяти для локальных участков рабочих массивов. В результате значительно ускоряется работа программы, что, в свою очередь, позволяет на несколько порядков уменьшить продолжительность обработки изображений в методах цифровой трассерной визуализации с построением и оптимизацией реконструированного кадра. Этот метод обеспечивает большую точность определения скоростей, чем стандартная кросс-корреляция, и позволяет использовать менее дорогое и более доступное оборудование для регистрации изображений трассеров.

Дальнейшее повышение быстродействия программ обработки цифровых изображений, основанной на предложенном методе, может быть реализовано на основе реализации параллельных вычислений. При этом наиболее перспективным представляется использование параллельных вычислений на графических процессорах, в частности, на программной платформе CUDA.

Библиографический список

1. Willert C.E. Digital particle image velocimetry / C.E. Willert, M. Charib // Exp. Fluids. − 1991. − V. 10. − P. 181 − 193.

2. Raffel M. Particle Image Velocimetry. A practical guide / M.Raffel, C.E. Willert, S.T. Wereley, J. Kompenhans. − Springer, 2007. − 448 pp.

3. Токарев М.П. Разработка алгоритмов и программного обеспечения для обработки изображений в методах цифровой трассерной визуализации: дис. ... канд. техн. наук: 05.13.18 / М.П. Токарев. – Новосибирск, 2010. – 190 с.

4. Мочалин А.Е. Информационная технология обработки трассерных изображений, основанная на построение реконструированного кадра / А.Е. Мочалин // Актуальные вопросы теплофизики и физической гидрогазодинамики: XI − я Всероссийская школа – конференция молодых ученых: тезисы докладов.– Новосибирск, 2010.– С. 66.

5. Мочалин А.Е. Информационная технология трассерной визуализации, основанная на оптимизации реконструированного изображения / А.Е. Мочалин // Науково - технічний журнал «Радіоелектронні і комп'ютерні системи». – 2010.– №4.– С. 174 − 178.

6. Huang H.T. Limitation and improvement of PIV/ H.T. Huang, H.E. Fiedler, J.J. Wang// Exp. in Fluids.– 1993.– Vol. 15, №4–5.– P. 263 – 273.

7. Рихтер Д. Windows для профессионалов: пер. с англ.: Создание эффективных Win32 − приложений с учетом особенностей 64 − разрядной версии Windows / Д. Рихтер.− 4 - е изд.− СПб.: Питер4 М.: «Русская редакция», 2001.− 723 с.

Рекомендована к печати к.т.н., проф. Паэрандом Ю.Э