МНОГОПОТОЧНАЯ РЕАЛИЗАЦИЯ ОБРАТНОГО КОМПОЗИЦИОННОГО АЛГОРИТМА ГАУССА-НЬЮТОНА

БересневА.П.
Научный руководитель: БолотоваЮ.А.
Национальный исследовательский Томский политехнический университет
apb3@tpu.ru

Введение

В процессе разработки и изготовления новых деталей необходимо провести их испытание и тестирование. Первоначальным этапом при описании деформационного состояния объекта является определение векторов смещений (оптический поток) ключевых точек на серии последовательных изображений поверхности материалов. Обратный композиционный алгоритм Гаусса-Ньютона (IC-GN) был первоначально предложен Бэйкером и Мэтьюсом в 2001-м году как эффективный метод стабилизации изображения [1]. Затем, Пан и др. [2] использовали этот алгоритм, скомбинировав с ZNSSD критерием, как субпиксельный метод нахождения оптического потока. Данный алгоритм позволяет с высокой точностью найти оптический поток, но обладает существенным недостатком в плане скорости работы. В данной работе описан способ многопоточной реализации.

Описание однопоточного алгоритма

IC-GN рассматривает прямоугольную область на первом (исходное) изображении и находит соответствующую ей область на втором (целевое) изображении последовательности. Для того чтобы найти оптический поток, изображение разбивается на сетку небольших областей в пределах которых смещение принимается одинаковым. Для каждой из областей IC-GN применяется независимо.

Формально алгоритм состоит из двух этапов:

1. целочисленное смещение;
2. субпиксельном смещение.

На первом этапе применяется свертка Reference и Target областей с использованием быстрого преобразования Фурье (БПФ) с нормированной суммой разности квадратов (ZNCC).

$$C_{zncc} = \frac{\sum_i \acute{R}_i \acute{T}_i}{\sqrt{\sum_i (\acute{R}_i)^2 \sum_i (\acute{T}_i)^2}},$$

где i – номер пикселя в области, $\acute{R}_i = R_i - R_m$, а $\acute{T}_i = T_i - T_m$, R_m и T_m – среднее значение яркости пикселей в областях на изображении. Далее, смещение находится как максимум ZNCC.

На втором этапе целочисленное смещение уточняется. На этом этапе минимизируется ошибка критерия нормированной суммы разности. Так как вычисляется субпиксельное смещение, то на данном этапе яркость пикселей с вещественными координатами интерполируется с использованием бикубической интерполяции [3]. Вектора параметров деформации и его вектор ошибки соответственно:

$$p = \left[u, u_x, u_y, v, v_x, v_y\right]^T,$$

$$\Delta p = \left[\Delta u, \Delta u_x, \Delta u_y, \Delta v, \Delta v_x, \Delta v_y\right]^T.$$

Вектор ошибки обновляется с использованием:

$$\Delta p = H^{-1} \sum_\xi \left\{\left[\nabla R(\psi + \xi) \frac{\partial W}{\partial p}\right] \left[\frac{R_n}{T_n} \acute{T}[\psi + W(\xi, p)] - \acute{R}(\psi + \xi)\right]\right\} (1)$$

Итеративно процесс алгоритма описывается в четыре шага:

построить целевую область $\acute{T}[\psi + W(\xi, p)]$, используя интерполяцию;

вычислить $\sum_\xi \left\{\left[\nabla R(\psi + \xi) \frac{\partial W}{\partial p}\right] \left[\frac{R_n}{T_n} \acute{T}[\psi + W(\xi, p)] - \acute{R}(\psi + \xi)\right]\right\}$ из выражения (1) с текущим значением p;

вычислить инкрементный вектор Δp, используя выражение (1);

обновить функцию $W(\xi, p)$.

Эти четыре шага повторяются итеративно, пока одно из следующих условий не будет выполнено:

$$|\Delta p| = \sqrt{(\Delta u)^2 + (\Delta v)^2} < 0.000001;$$

достигнуто максимальное количество итераций [1].

Описание многопоточного алгоритма

Узкое место алгоритма состоит в том, что на каждой итерации необходимо субпиксельно интерполировать второе изображение. Для реализации на GPU используется библиотека OpenCL. Реализация состоит из ведущего приложения и совокупности ядер (kernel). Кернел представляет собой набор инструкций, написанных на языке OpenCL, которые возможно исполнять на различных устройствах, таких как центральный процессор и графический процессор. Так как графический процессор состоит из большого числа исполнительных устройств (work-item), то ядра можно запустить на них параллельно.

Обобщенно алгоритм выполнения многопоточных программ состоит из следующих этапов:

Инициализация – ведущая программа получает информацию о аппаратном обеспечении, на котором она была запущена и создает контекст исполнения, основываясь на текущей платформе.

На этом этапе так же создается очередь исполнения ядер;

Перенос объектов с основной памяти на память исполнительных устройств;

Создание ядер и их запуск на исполнение – создаются копии ядер для каждого исполнительного устройства, которые затем выполняются параллельно;

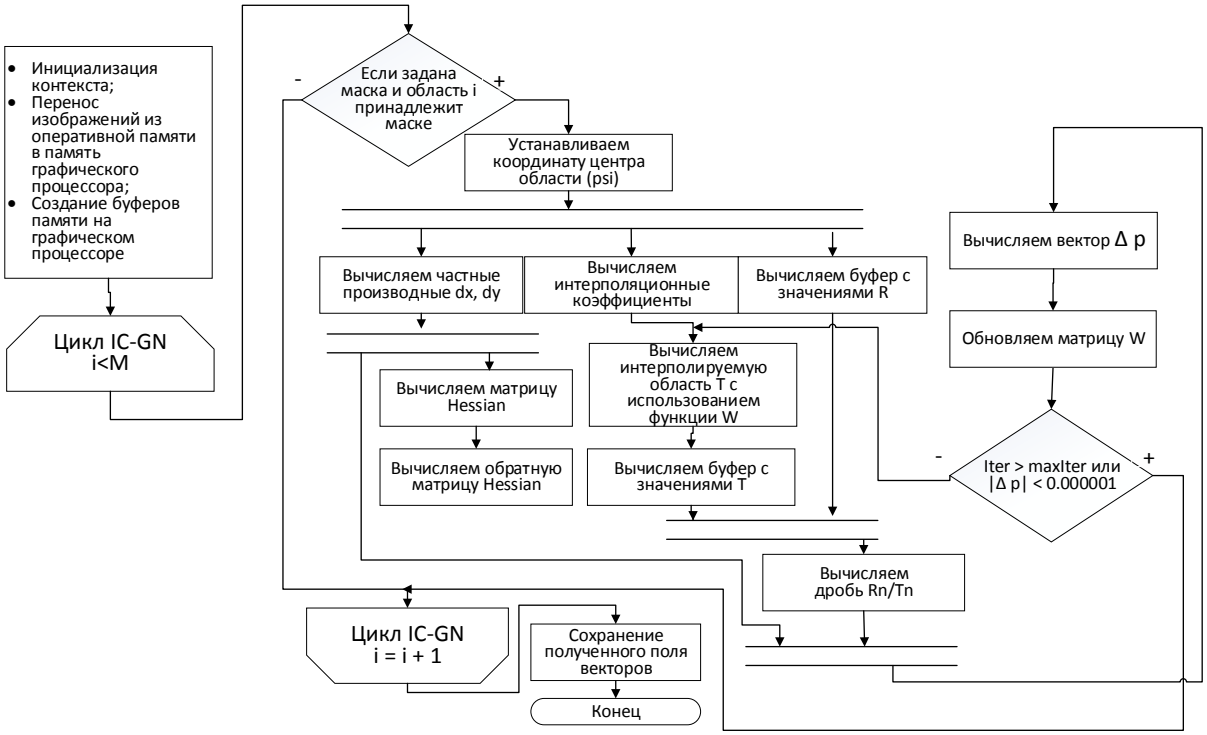
Ожидание завершения исполнения параллельного кода и перенос результатов вычислений с памяти исполнительных устройств на основную память;

Освобождение ресурсов и завершение работы программы [4].

Дополнительного увеличения скорости обработки можно достичь через параллельное исполнение команд помимо распараллеливания обработки данных возможна. Тем самым, будут задействованы неиспользуемые исполнительные устройства. Что должно привести к дополнительному увеличению скорости обработки данных.

Так как для БПФ уже существуют готовые реализации [5], то необходимо реализовать параллельный вариант второй части алгоритма.

В целом, многопоточная версия алгоритма отличается от однопоточной тем, что используются параллельные методы обработки.Ниже представлена блок-схема разработанной многопоточной версии алгоритма.



Блок-схема многопоточного алгоритма

Заключение

Разработанная многопоточная версия алгоритма имеет преимущество в скорости за счет многопоточных вычислений. Однако, негативным фактором при многопоточной работе на GPU является медленная скорость переноса буферов с основной памяти на память устройства.

В качестве дальнейших перспектив возможно улучшение существующего алгоритма, оптимизация кода и внедрение новых изменений.

Список использованных источников

1. Jiang Z, Kemao Q, Miao H, Yang J, Tang L. Path-independent digital image correlation with high accuracy, speed and robustness. Opt Laser Eng 2015;65: С. 93 – 102.

2. BakerS.,Matthews I.Equivalence and efficiency of image alignment algo- rithms. In:Proceedings of the IEEE Conferenceon Computer Vision and Pattern Recognition,vol.56;2001. С. 1090–97

3. «Cubic interpolation» [Электронный ресурс]. (1.06.16) Режим доступа: http://www.paulinternet.nl/?page=bicubic

4. L. Zhang et al. High accuracy digital image correlation powered by GPU-based parallel computing. Optics and Lasers in Engineering 69 (2015) С. 7–12

5. «OpenCL™ Optimization Case Study Fast Fourier Transform» [Электронныйресурс]. (1.06.16) Режим доступа: http://developer.amd.com/resources/articles-

whitepapers/opencl-optimization-case-study-fast-
fourier-transform-part-1/