1.3. РЕАЛИЗАЦИЯ ГЕОМЕТРИЧЕСКОГО ПРЕОБРАЗОВАНИЯ ОТРАЖЕНИЯ ДЛЯ ПРОТОТИПА ВЫСОКОПРОИЗВОДИТЕЛЬНОГО ГЕОМЕТРИЧЕСКОГО ЯДРА

Егунов В.А.[1], Филимонов О.Ю.[2]
[1]ВолгГТУ, г. Волгоград
[2]ООО «Сингулярис Лаб», г. Волгоград

В данной работе представлены результаты по оптимизации вычислений аффинных преобразований для полигональных моделей, которые широко используются в геометрических ядрах САПР. Рассмотрены основные форматы хранения данных полигональных моделей, приводится матричное представление полигональных моделей. Рассматриваются различные варианты выполнения преобразования отражения над полигональными моделями, описывается процесс векторизации вычислений. Приводится сравнение различных вариантов выполнения.

Введение

Существует определённый спектр задач, для решения которых требуется использование вычислительной геометрии. Примером таких задач может служить вычисление объёма произвольного трёхмерного тела, булевы операции над полигональными моделями, триангуляция многоугольника или более простые задачи по преобразованию тел в пространстве. Такие задачи встречаются в 3D проектировании, а именно в CAD, CAM и CAE системах, в робототехнике, ГИС и т.д.

В докладе, подготовленном коллективом Института статистических исследований и экономики знаний НИУ ВШЭ [Абдрахманова, 2022] отмечается, что как в последние годы, так и в настоящее время, в нашей стране успешно развивались ключевые элементы цифровой экономики. Более того, отмечаются устойчивые тренды цифровой трансформации экономики и социальной сферы. Базой для выполнения данной трансформации являются, в том числе, программные продукты для проектирования и компьютерного инжиниринга – CAD, CAM, CAE и пр. [Полянсков, 2013].

Для таких задач ключевой параметр – это оценка сложности используемых алгоритмов [Дешко, 2021]. Так, разница между алгоритмом с квадратичной и линейно-логарифмической сложностью может выливаться в дополнительные часы или дни вычислений. Для оптимизации времени работы алгоритма, если отсутствует альтернатива в виде идентичного алгоритма с более низкой сложностью, можно использовать различные методы – распараллеливание, векторизация [Егунов, 2020], кэш-оптимизация [Егунов, 2019], вычисление на GPU и т.д.

В данной работе рассматривается эффективность реализаций аффинного преобразования отражения для полигональных моделей.

Геометрические ядра.

Как правило, для решения вышеописанных задач используется геометрическое ядро, если речь идёт о САПР, или его более лёгкий аналог. Геометрическое ядро – набор библиотек с программным интерфейсом, с помощью которого можно пользоваться функциями геометрического моделирования. Наиболее известные геометрические ядра – это, к примеру, Parasolid, Open CASCADE. Последнее геометрическое ядро имеет открытый исходный код, так что для него существует ряд API для различных языков программирования. Из отечественных разработок можно отметить ядро C3D, которое лежит в основе Компас-3D, и RGK (Russian Geometric Kernel) [Стешина, 2015], в процессе разработки которого были написаны несколько статей [Шаповалов, 2013], также задействованы различные компании и университеты [Filimonov, 2021]. Более подробно о самих геометрических ядрах и их роли в мире САПР можно узнать здесь [Цымбал, 2022].

Одной из важных характеристик геометрических ядер является возможность работать с разным представлением данных, что определяет возможности ядра по чтению и загрузке моделей, хранящихся в различных геометрических форматах, Речь здесь может идти как о форматах, разработанных специально для конкретной САПР, так и об общедоступных форматах. Геометрические форматы первой группы, как правило, содержат не только саму модель, но и сцену, куда включается освещение, тени, другие модели и т.д. Из общедоступных форматов можно перечислить следующие: STL, OBJ, COLLADA, FBX, STEP. Стоит отметить, что наиболее популярными являются STL [Stroud, 2000] и COLLADA. У некоторых общедоступных форматов существует проблема универсальности, так как для них нет точного описания спецификации.

Матричное описание полигональной сетки

Полигональная модель описывается набором полигонов и нормалей [Гонахчян, 2014]. В качестве полигона может выступать любой плоский многоугольник, однако на практике используется треугольный полигон. Так, полигональная модель с N полигонами содержит N нормалей и N * 3 вершин. Таким

образом, полигональную модель можно представить в виде матриц, показанных на рисунке 1 (широкая матрица) и рисунке 2 (высокая матрица).

Оба матричных представления имею свои достоинства и недостатки, поэтому в разных алгоритмах будет использоваться представление, описание которого по структуре и по расположению данных в памяти вычислительной машины является более предпочтительным для рассматриваемого алгоритма. К примеру, для широкого представления в кэш попадёт массив однотипных координат (x, y или z), а для высокого представления в кэше будет находиться набор вершин, что для линейных преобразований таких, как вращение и отражение, может значительно снизить время выполнения алгоритма.

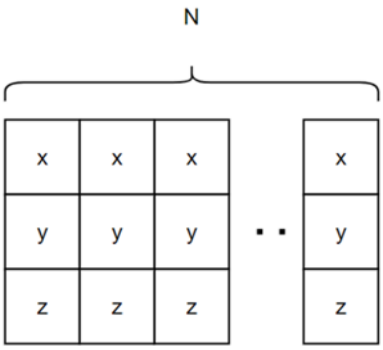

Рисунок 1 Представление в виде матрицы 3xN, N – число вершин

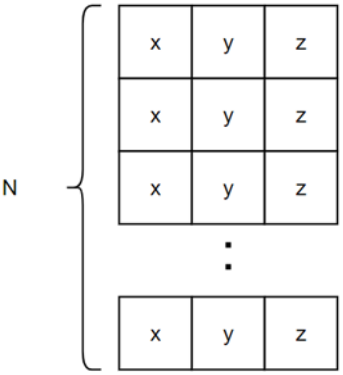

Рисунок 2 Представление в виде матрицы Nx3, N – число вершин

Непосредственно само полигональное описание 3D-модели может использоваться во многих областях: начиная от 3D-творчества в Blender и заканчивая использованием в 3D-печати [Andreev, 2021].

Оптимизация преобразования отражения

В данной работе в качестве оптимизируемой операции рассматривалось преобразование отражение. Данная операция выполняется путем применения матрицы отражения (оператор Хаусхолдера) к набору векторов, которые могут быть записаны в виде матрицы 3xN или Nx3.

$$R = H(n)\bar{x} \quad 1)$$

где H(n) – матрица Хаусхолдера (2), n – нормаль к гиперплоскости, от которой происходит отражение, $\bar{x}$ – отражаемый вектор.

$$H(n) = I - 2nn^T \quad 2)$$

В качестве отражаемого объекта может выступать и набор векторов. Двустороннее преобразование Хаусхолдера часто используется для приведения исходной матрицы к Хессенберговой или двухдиагональной (ленточной) форме. В качестве стороннего используемого метода рассматривается функция LAPACKE_?larfx из пакета Intel MKL [Intel, 2015].

Векторизация вычислений

Для повышения производительности операция отражения была векторизована. Для векторизации использовались интринсики (SSE) [Harris, 2012]. Поскольку работа осуществляется с вершинами в трёхмерном пространстве, то использование регистров xmm усложняется тем, что для типа float и double в них можно поместить по 4 и 2 элемента соответственно. Одна вершина описывается 3 точками. Это значит, что для типов float и double требуется одновременная обработка 4 и 2 вершин соответственно для заполнения векторов. Как будет показано далее, наиболее предпочтительным для векторизации является формат Nx3, опишем векторизацию алгоритма с использование данного формата. На рисунке 3 показано распределение координат 4 вершин в трёх регистрах XMM для типа float в представлении Nx3

Рисунок 3. – Распределение координат 4 вершин в трёх регистрах XMM для типа float в представлении Nx3

Матрица Хаусхолдера симметрична, поэтому для описания алгоритма векторизации она будет описываться следующим образом:

$$H = \begin{bmatrix} a & f & e \\ f & b & g \\ e & g & c \end{bmatrix} \quad 3)$$

Всего требуется вычислить 3 квартета значений, что разбивает один проход на 3 отдельных шага, которые описаны на рисунках 4, 5 и 6. Вектора множителей (к примеру [a, f, e, a]), образованные из матрицы H, циклично сдвигаются на каждом шаге влево так, что последним элементом становится новый первый, то есть вектор [a, f, e, a] на первом шаге станет [f, e, a, f] на втором. Аналогичный сдвиг будет выполнен и на шаге 3. Выполнение векторизации для типа double осуществляется аналогично, за тем исключением формирования вектора множителей.

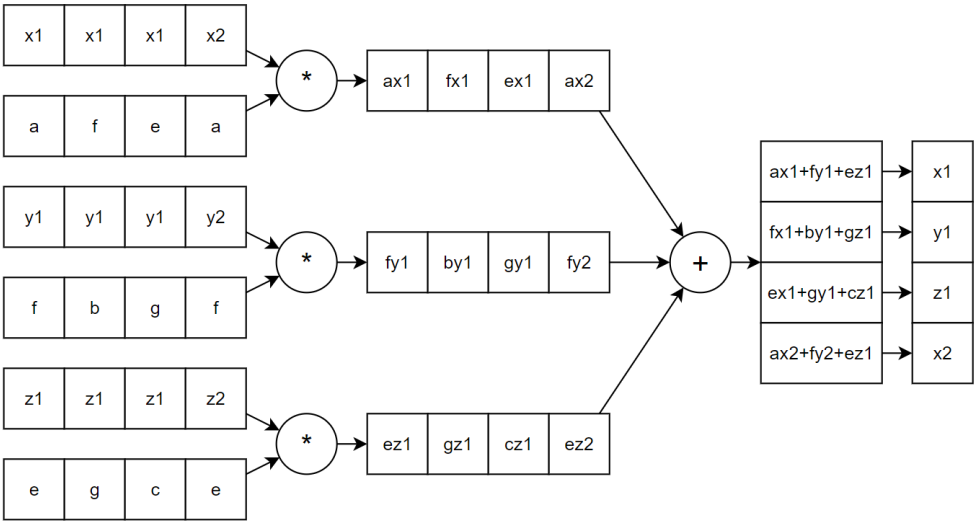

Рисунок 4 – Схема выполнение 1/3 части шага для векторизованной операции отражения. Тип данных – float

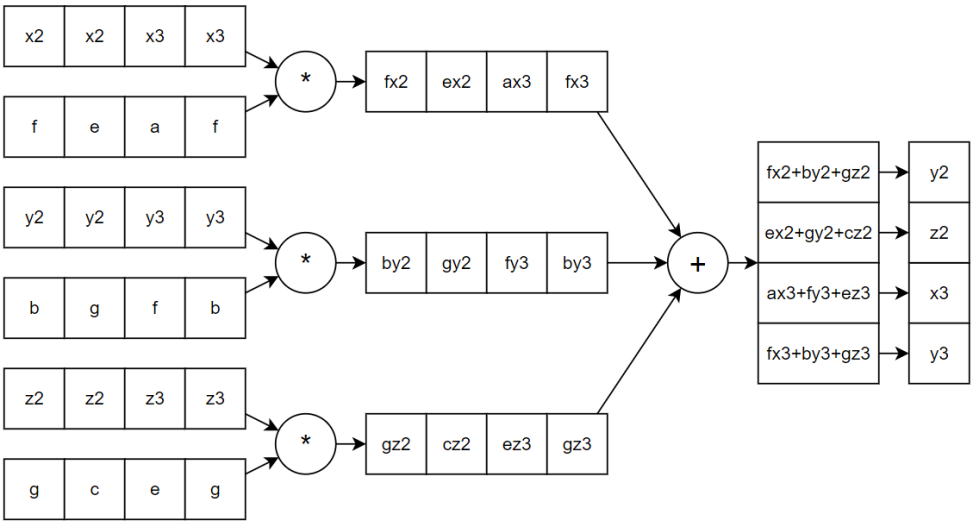

Рисунок 5 – Схема выполнение 2/3 части шага для векторизованной операции отражения. Тип данных – float

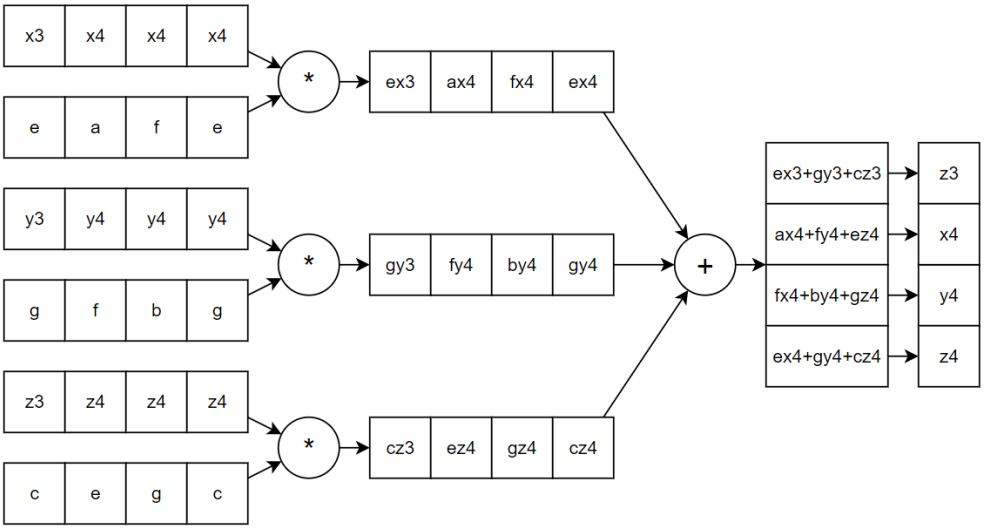

Рисунок 6 – Схема выполнение 3/3 части шага для векторизованной операции отражения. Тип данных – float

Результаты исследования

Результаты вычислительных экспериментов представлены на рисунках 7, 8, 9. На рисунках представлены обозначения в формате name_format_type. Здесь name – название алгоритма, format – формат хранения данных, type – тип используемых данных. Name принимает значения:

- standart – матричная форма реализации отражения Хаусхолдера;
- hessenberg – векторная форма реализации отражения Хаусхолдера;
- vectorized_standart – векторизованный вариант матричной формы реализации отражения Хаусхолдера;
- vectorized_hessenberg – векторизованный формат векторной формы реализации отражения Хаусхолдера;
- mkl – реализация Intel MKL.

Format может принимать значения 3xN и Nx3, сами форматы описаны выше. Type может принимать значения double и float, что соответствует типам используемых данных (программы были написаны на C++).

Например, standart_nx3_float означает реализацию матричной формы реализации отражения Хаусхолдера, формат хранения – «высокая матрица», тип данных – float.

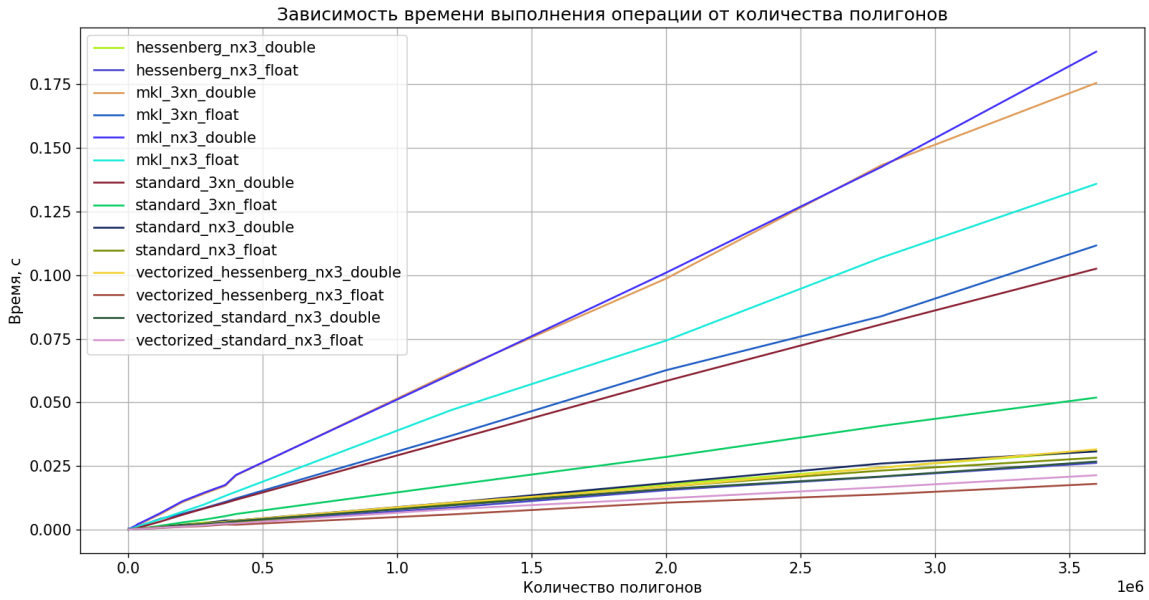

Рисунок 7 – Зависимость времени выполнения операций от количества полигонов

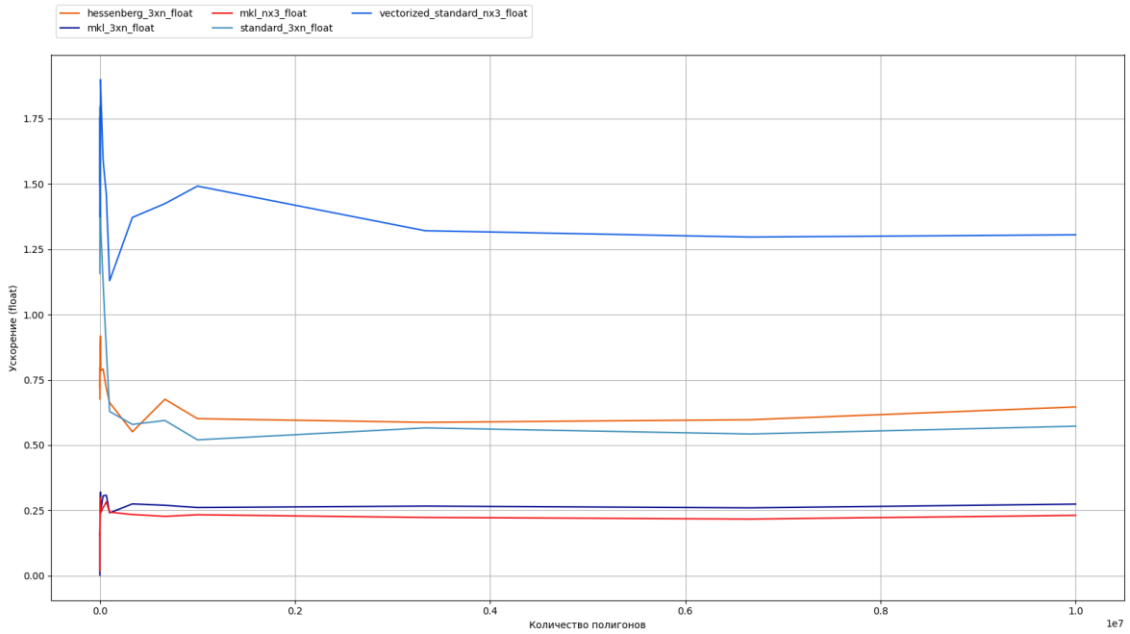

Рисунок 8 – Зависимость ускорения от количества полигонов. Тип float

Анализируя зависимости на рисунке 7, можно сделать следующие выводы:
- в общем случае формат хранения Nx3 является более предпочтительным;
- векторная форма реализации отражения Хаусхолдера является более предпочтительной;
- векторизация уменьшает общее время выполнения программы для всех методов;
- функция библиотеки Intel MKL в данном случае отработала плохо.

Самым быстрым алгоритмом является vectorized_hessenberg_nx3 (в обеих группах алгоритмов – float и double), самым медленным – mkl_3xn.

Далее было проанализировано ускорение полученных программ по отношению к эталонному методу – standart_nx3_[float|double].

Видно, что быстрее работает только векторизованная версия эталонного метода, все остальные реализации проигрывают, причем MKL-реализация проигрывает значительно.

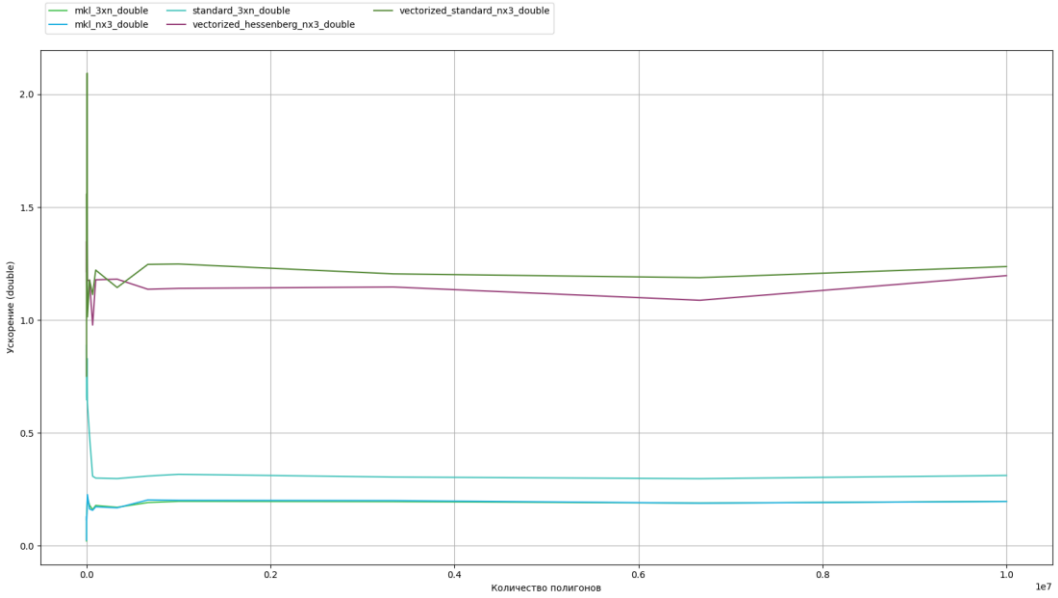

Рисунок 9 – Зависимость ускорения от количества полигонов. Тип double

Здесь также преимущество у векторизованных Nx3-схем, остальные, включая MKL-реализации, показывают гораздо худшие результаты.

В целом можно сказать, что наблюдаются те же тренды, что и на рисунке 7, при этом с некоторого количества полигонов ускорение начинает вести себя линейно. Однако на практике нечасто используются полигональные модели с порядком 10^6. Поэтому имеет смысл проанализировать ускорения на малом количестве полигонов (до 10000).

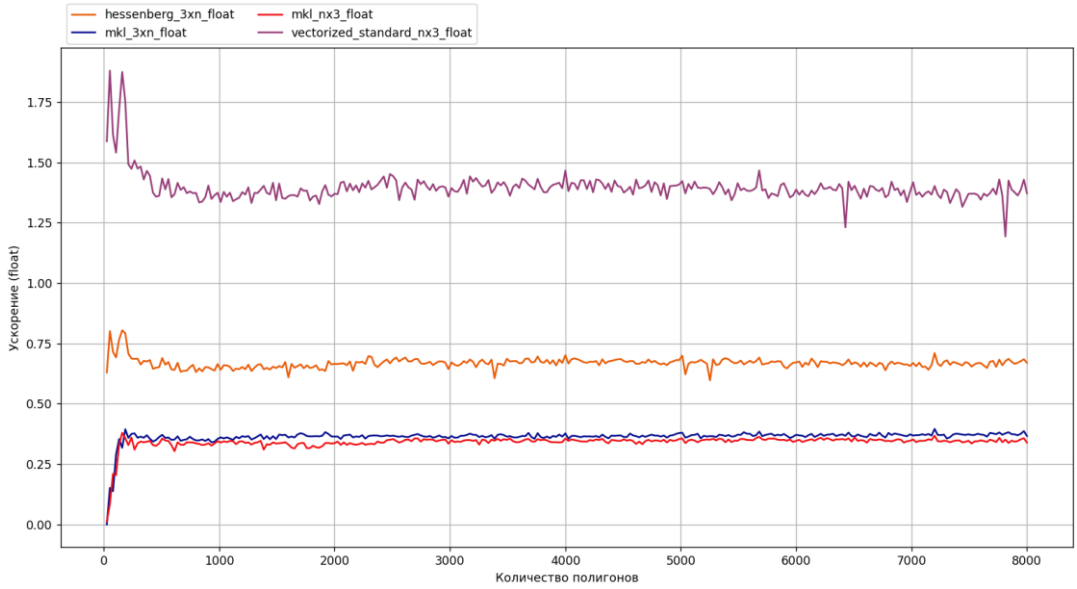

Рисунок 10 – Зависимость ускорения от количества полигонов. Тип float

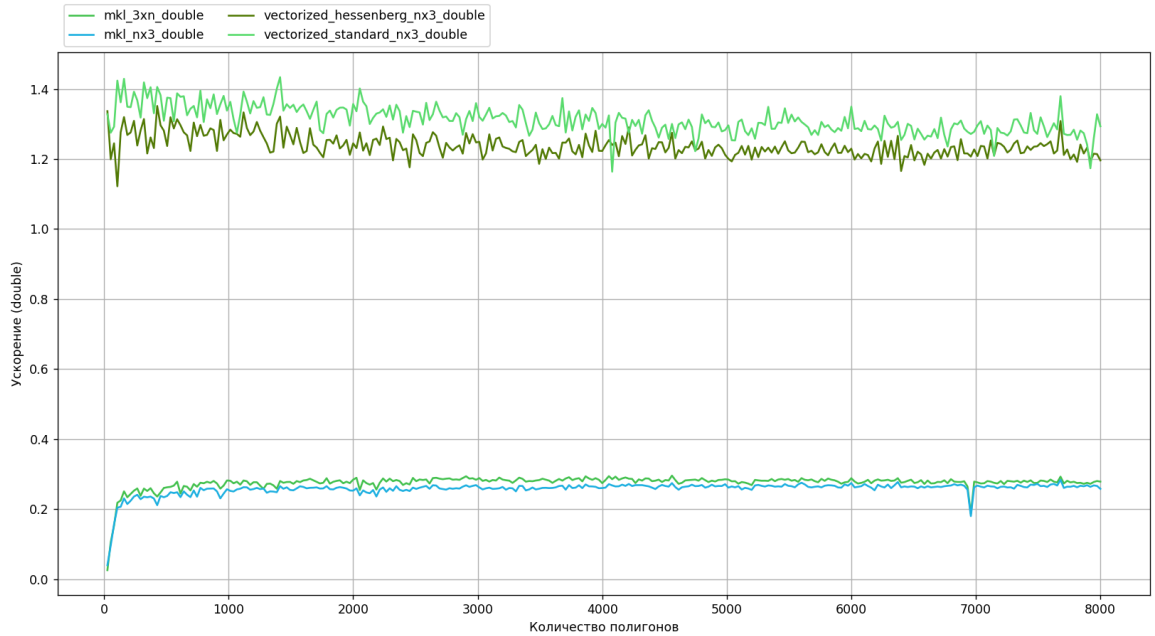

Рисунок 11 – Зависимость ускорения от количества полигонов. Тип double

Из графиков 10 и 11 видно, что ускорение на малом количестве полигонов также ведёт себя линейно.

Выводы

По результатам оптимизации операции отражения можно заключить, что оптимальным форматом представления вершин является Nx3, его использование дает выигрыш во времени до 2,5 раз. Методы библиотеки Intel MKL оказались неэффективными, также неэффективными оказались методы, основанные на обработке полигональной модели в формате 3xN. Гораздо более предпочтительным оказалось использование векторной формы реализации преобразования отражения, при этом векторизация вычислений дает существенный, до 2 раз, прирост производительности.

Так как данные о вершинах полигональных моделей, как правило, хранятся с использованием внешней индексации, полученные методы могут быть использованы для выполнения преобразования отражения в геометрических ядрах. Направление дальнейших исследований будет связано с исследованием реализации других базовых геометрических операций, таких как параллельный перенос, скалирование, вращение, отражение, сдвиг и т.д., для реализации на полигональных моделях.

Литература

1. Абдрахманова Г. И., Васильковский С. А., Вишневский, М. А. Гершман, Л. М. Гохберг и др. Цифровая трансформация: ожидания и реальность: докл. к XXIII Ясинской (Апрельской) междунар. науч. конф. по проблемам развития экономики и общества, Москва, 2022 г. [Текст] / рук. авт. кол. П. Б. Рудник; Нац. исслед. ун-т «Высшая школа экономики». — М.: Изд. дом Высшей школы экономики, 2022. — 221 с. — ISBN 978-5-7598-2658-3 (в обл.). — ISBN 978-5-7598-2468-8 (e-book).
2. Гонахчян В.И. Обзор методов упрощения полигональных моделей на графическом процессоре. Труды Института системного программирования РАН. 2014. Т. 26. № 2. С. 159-174.
3. Дешко И.П., Цветков В.Я. Оценка сложности алгоритма. Славянский форум. 2021. № 3 (33). С. 38-49.
4. Егунов, В.А., Андреев А.Е. Векторизация алгоритмов выполнения собственного и сингулярного разложений матриц с использованием преобразования Хаусхолдера / Прикаспийский журнал: управление и высокие технологии. – 2020. – № 2 (50). – С. 71-85.
5. Егунов, В.А. Кэш-оптимизация процесса вычисления собственных значений на параллельных вычислительных системах / В.А. Егунов // Прикаспийский журнал: управление и высокие технологии. – 2019. – № 1 (45). – С. 154-163.
6. Полянсков Ю.В. Интеграция САПР-, PDM-, ERP-систем в единое информационное пространство производственного предприятия. / Ю.В. Полянсков, А.С. Кондратьева, М.С. Черников, А.А. Блюменштейн. – Известия Самарского научного центра Российской академии наук, т. 15, № 4(3), 2013.
7. Стешина Д.А., Кочан И.Н. T-FLEX PLM: импортозамещение – это реально! Автоматизация в промышленности. 2015. № 1. С. 47-49.
8. Цымбал Д.В., Юров А.Н., Степенко С.А. Анализ геометрических ядер. «Информатика: проблемы, методы, технологии», 2022, стр. 1284–1291.

9. Шаповалов О.В., Сергеев Е.С., Чалышев В.С., Катаев А.В., Крыжановский Д.И., Андреев А.Е. Вопрос распараллеливания в разработке ядра геометрического моделирования. Научный сервис в сети Интернет: все грани параллелизма, 2013.
10. Andreev, A., Andreeva, M., Drobotov, A., Shmeleva, A., Denisov, M. Development of a Discrete Slicer for Additive Manufacturing. Communications in Computer and Information Science, 2021, 1448 CCIS, pg. 267–281.
11. Filimonov, O.Y., Egunov, V.A., Nesterenko, E.N. Constructing Equidistant Curve for Planar Composite Curve in CAD Systems. Communications in Computer and Information Science, 2021, 1448 CCIS, pg. 296–309.
12. Harris D., Harris S. Digital Design and Computer Architecture // 2nd Edition. NY, Morgan Kaufmann, 2012. 712 p.
13. Intel. Math Kernel Library. https://software.intel.com/en-us/intel-mkl. (2015).
14. Stroud I., Xirouchakis P.C. STL AND EXTENSIONS. Advances in Engineering Software. 2000. Т. 31. № 2. С. 83–95.

References in Cyrillics

1. Abdrahmanova G. I., Vasil'kovskij S. A., Vishnevskij, M. A. Gershman, L. M. Gohberg i dr. Cifrovaya transformaciya: ozhidaniya i real'nost': dokl. k XXIII YAsinskoj (Aprel'skoj) mezhdunar. nauch. konf. po problemam razvitiya ekonomiki i obshchestva, Moskva, 2022 g. [Tekst] / ruk. avt. kol. P. B. Rudnik; Nac. issled. un-t «Vysshaya shkola ekonomiki». — M.: Izd. dom Vysshej shkoly ekonomiki, 2022. — 221 s. — ISBN 978-5-7598-2658-3 (v obl.). — ISBN 978-5-7598-2468-8 (e-book).
2. Gonahchyan V.I. Obzor metodov uproshcheniya poligonal'nyh modelej na graficheskom processore. Trudy Instituta sistemnogo programmirovaniya RAN. 2014. T. 26. № 2. S. 159-174.
3. Deshko I.P., Cvetkov V.YA. Ocenka slozhnosti algoritma. Slavyanskij forum. 2021. № 3 (33). S. 38-49.
4. Egunov, V.A., Andreev A.E. Vektorizaciya algoritmov vypolneniya sobstvennogo i singulyarnogo razlozhenij matric s ispol'zovaniem preobrazovaniya Hauskholdera / Prikaspijskij zhurnal: upravlenie i vysokie tekhnologii. – 2020. – № 2 (50). – C. 71-85.
5. Egunov, V.A. Kesh-optimizaciya processa vychisleniya sobstvennyh znachenij na parallel'nyh vychislitel'nyh sistemah / V.A. Egunov // Prikaspijskij zhurnal: upravlenie i vysokie tekhnologii. – 2019. – № 1 (45). – C. 154-163.
6. Polyanskov YU.V. Integraciya SAPR-, PDM-, ERP-sistem v edinoe informacionnoe prostranstvo proizvodstvennogo predpriyatiya. / YU.V. Polyanskov, A.S. Kondrat'eva, M.S. CHernikov, A.A. Blyumenshtejn. – Izvestiya Samarskogo nauchnogo centra Rossijskoj akademii nauk, t. 15, № 4(3), 2013.
7. Steshina D.A., Kochan I.N. T-FLEX PLM: importozameshchenie – eto real'no! Avtomatizaciya v promyshlennosti. 2015. № 1. S. 47-49.
8. Cymbal D.V., YUrov A.N., Stepenko S.A. Analiz geometricheskih yader. «Informatika: problemy, metody, tekhnologii», 2022, str. 1284–1291.
9. SHapovalov O.V., Sergeev E.S., CHalyshev V.S., Kataev A.V., Kryzhanovskij D.I., Andreev A.E. Vopros rasparallelivaniya v razrabotke yadra geometricheskogo modelirovaniya. Nauchnyj servis v seti Internet: vse grani parallelizma, 2013.

Ключевые слова

Триангуляция, полигональные модели, матричное представление, оптимизация вычислений, интринсики, аффинные преобразования, преобразование Хаусхолдера, векторизация

Егунов Виталий Алексеевич – доцент кафедры «ЭВМ и системы»
Волгоградского государственного технического университета, Волгоград, Россия,
ORCID: 0000-0001-9087-3275, vegunov@mail.com

Филимонов Олег Юрьевич – инженер-программист ООО «Сингулярис Лаб», Волгоград, Россия
filimonovoleg@bk.ru

Keywords

Triangulation, polygonal models, matrix representation, optimization of calculations, intrinsics, affine transformations, Householder transformation, vectorization

Vitaly Egunov, Oleg Filimonov, Implementation of geometric reflection transformation for a prototype of a high-performance geometric kernel

DOI: 10.34706/DE-2023-01-03
JEL classification C02 – Математические методы; M15 Управление информационными технологиями.

Abstract. This paper presents the results on optimization of calculations of affine transformations for polygonal models, which are widely used in geometric CAD kernels. The main formats of data storage of polygonal models are considered, a matrix representation of polygonal models is given. Various variants of the reflection transformation over polygonal models are considered, the process of vectorization of calculations is described. A comparison of different execution options is given.