1. **Аннотация**
2. **Введение**

Дискретные преобразования – один из важнейших и широко использующихся инструментов в алгоритмах цифровой обработки сигналов. Дискретные преобразования как дискретный аналог непрерывных интегральных преобразований позволяют получать для входных данных различные представления, обычно называющихся спектром. Наиболее известным и распространенным преобразованием является дискретное преобразование Фурье (DFT). В частности, преобразование Фурье используется в таких важнейших прикладных приложениях, как сжатие медиа-данных (MP3, JPEG) [1] [2], спектральном анализе геоданных [3] и т.д.

Еще одним хорошо известным преобразованием является дискретное преобразование Хартли (DHT). В отличие от преобразования Фурье, преобразование Хартли выполняется только в действительной области, что может быть особенно выгодно при обработке больших данных.

Одномерное дискретное преобразование Хартли определяется, как:

|  |  |
| --- | --- |
|  | (1) |

Помимо обработки одномерных сигналов существует огромное количество задач по обработке данных большей размерности. Например, в двухмерном случае используются алгоритмы компьютерного зрения, которые обрабатывают данные, представленные в виде матриц – от пользовательских фотографий [4] до томограмм, полученных при помощи КТ [5].

1. **Вычисление дискретного 3D преобразования Хартли**
   1. **3D преобразование как последовательность 1D преобразований**

Классическим походом при вычислении различных дискретных 3D преобразований, в том числе, преобразования Хартли, является представление 3D преобразования в виде последовательности независимо вычисляемых 1D преобразований. Такой подход при вычислении позволяет производить тривиальное распараллеливание [6]. На Рисунке 1 изображена схема вычисления преобразования в 2D случае: для вычисления 2D преобразования Хартли квадратного изображения разрешением сначала выполняется 1D преобразований вдоль оси , а потом 1D преобразований вдоль оси (либо наоборот).

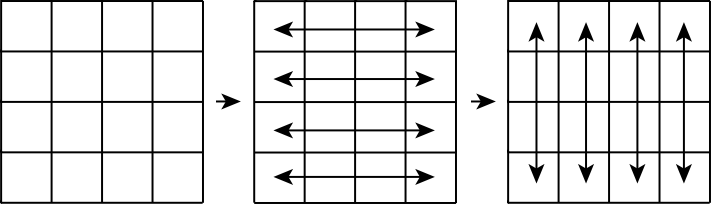


Рисунок 1. Идея нахождения 2D преобразования Хартли путём вычисления 1D преобразований Хартли вдоль оси Х, а после - вдоль оси Y.

В случае вычисления преобразования Фурье этот подход становится возможным из-за разделимости ядра преобразования. При вычислении преобразования Хартли применяются подстановки, предложенные Bracewell и Hao в работах [7] и [8].

* 1. **Быстрые преобразования**

В большинстве приложений одномерные преобразования вычисляются с помощью быстрых алгоритмов [9]. В случае последовательного вычисления преобразования Хартли по определению потребуется арифметических операций. Однако быстрые алгоритмы, например, алгоритмы Cooley-Tukey, получают тот же результат за операций.

* 1. **Распараллеливание**

Современный подход разработки алгоритмов требует их распараллеливания. Алгоритмы Cooley-Tukey включают в себя несколько (а точнее ) простейших шагов вычисления, имеющих название «бабочка», каждый из которых внутри себя выполняется последовательно. При такой архитектуре алгоритм распараллеливается тем эффективнее, чем производительнее вычислительные ядра [статья гугл]. Таким образом, распараллеливание быстрых алгоритмов теоретически может сократить время выполнения до [статья], где – количество доступных ядер. Но их не может быть достаточно много для наиболее эффективного распараллеливания, если это мощные ядра CPU. В случае же GPU мы имеем достаточное количество для эффективного распараллеливания ядер, но не таких мощных, как на CPU, что так же является проблемой при распараллеливании.

1. **Матричное представление**

Другой способ представления DHT – выражение в виде умножения матрицы на вектор:

|  |  |
| --- | --- |
|  | (2) |

Где, – входной сигнал, – результат преобразования Хартли, – матрица в виде матрицы Вандермонда:

|  |  |
| --- | --- |
|  | (3) |

Матричный способ представления задачи дает возможность использовать лучшие практики распараллеливания на GPU [статья nvidia].

Важно отметить, что при серии вычислений DHT одного и того же размера матрица (3) вычисляется лишь один раз – это оказывает решающую роль при обработке ND данных.

Итак, для вычисления дискретных преобразований (в частности преобразования Хартли) таких специфических типов данных, как несколько сотен/тысяч изображений (3D томография, наборы изображений для обучения нейронных сетей и т.п.), различные 3D данные и данные больших размерностей (геология, геофизика и т.п.), мы предлагаем заменить использование быстрых одномерных преобразований Хартли на одномерные преобразования Хартли по определению, вычисленные на GPU.

1. **Результаты**

Мы провели серию экспериментов по обработке изображений с различными разрешениями: вычислили 2D преобразования Хартли для каждого из изображений с помощью вычисления

а) одномерных быстрых преобразований Хартли (распараллелено на CPU);

б) одномерных дискретных преобразований Хартли по определению (распараллелено на GPU).

* 1. **1D FHT на CPU**

Для вычисления одномерных преобразований Хартли мы реализовали классический прямой (не рекурсивный) алгоритм FHT [10] аналогичный алгоритму Cooley-Tukey для FFT. При вычислении 2D преобразования мы использовали OpenMP для вычисления 1D FHT. Так как каждое одномерное преобразование независимо, мы можем применить простейший подход: мы использовали #pragma omp parallel for перед циклом с вычислениями одномерных преобразований.

* 1. **1D DHT на GPU**

Как описано в Главе 4, мы свели задачу нахождения 1D DHT к умножению матрицы на вектор. В случае же вычисления 2D преобразования задача сводится к двум умножениям матриц (Рисунок 2) – вдоль оси Х и вдоль оси Y.

Мы использовали технологию CUDA для имплементации алгоритма. Задача умножения матриц – базовая задача в параллельных вычислениях на GPU, благодаря чему существует множество различных подходов к распараллеливанию. Мы же использовали простую реализацию, предложенную в [статья nvidia].

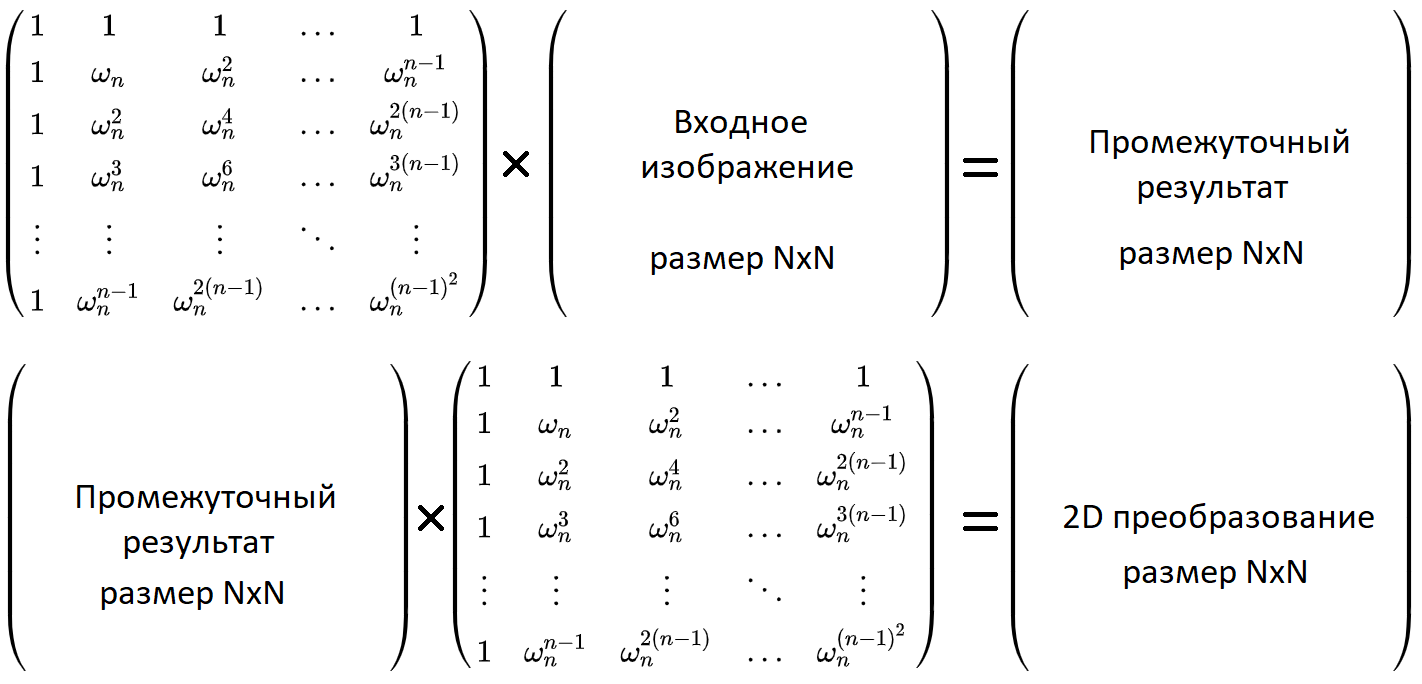


Рисунок 2. Нахождение дискретного 2D преобразования

Эксперименты по замерам производительности были проведены на компьютере Lenovo Ideapad 330 с процессором Intel Core i3-7020U 2.30 Ghz, 6ГБ ОЗУ и графическим ускорителем Nvidia GeForce MX150 2ГБ.

|  |  |  |
| --- | --- | --- |
| N | CPU | GPU |
| 50 | 7.26 | 1.95 |
| 100 | 12.87 | 2.48 |
| 200 | 25.31 | 3.63 |

Таблица 1. N изображений размером 1024x1024

|  |  |  |
| --- | --- | --- |
| N | CPU | GPU |
| 50 | 26.57 | 3.71 |
| 100 |  | 6.18 |
| 200 |  | 10.8 |

Таблица 1. N изображений размером 2048x2048

1. **Вывод**
2. **Список литературы**

|  |  |
| --- | --- |
| [1] | M. H. G., «A low-complexity design for an mp3 multi-channel audio decoding system,» *IEEE transactions on audio, speech, and language processing,* т. 20, № 1, pp. 314-321, 2011. |
| [2] | B. G. P.-G. F. Pasquini C., « IEEE International Workshop on Information Forensics and Security (WIFS),» в *Multiple JPEG compression detection by means of Benford-Fourier coefficients*, 2014. |
| [3] | S. M. C. J. Tahmasebi P., « MS-CCSIM: accelerating pattern-based geostatistical simulation of categorical variables using a multi-scale search in Fourier space,» *Computers & Geosciences,* pp. 75-88, 2014. |
| [4] | P. H. e. al., «Fcnn: Fourier convolutional neural networks,» *Joint European Conference on Machine Learning and Knowledge Discovery in Databases. – Springer,* pp. 786-798, 2017. |
| [5] | V. E. A. P. G. O. Arsenyev-Obraztsov S. S., «Proposals on 3D parallel edge-preserving filtration for x-ray tomographic digital images of porous medium core plugs,» *IOP Conference Series: Materials Science and Engineering. – IOP Publishing,* 2019. |
| [6] | M. M. K. D. E. Dalcin L., «Fast parallel multidimensional FFT using advanced MPI,» *Journal of Parallel and Distributed Computing,* pp. 137-150, 2019. |
| [7] | B. R. N. e. al., «Fast two-dimensional Hartley transform,» *Proceedings of the IEEE,* pp. 1282-1283, 1986. |
| [8] | B. R. N. Hao H., «A three-dimensional DFT algorithm using the fast Hartley transform,» *Proceedings of the IEEE,* pp. 264-266, 1987. |
| [9] | W. Z., «Fast algorithms for the discrete W transform and for the discrete Fourier transform,» *IEEE Transactions on Acoustics, Speech, and Signal Processing,* pp. 803-816., 1984. |
| [10] | H. H. S., « The fast Hartley transform algorithm,» *IEEE Transactions on Computers,* pp. 147-156, 1987. |