

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)**

Факультет программной инженерии и компьютерной техники

Мобильные системы компьютерного зрения

Лабораторная работа №3

Вариант №2: Сложение и вычитание двух изображений

Преподаватель: Денисов Алексей Константинович

Выполнили: Демьяновский Савелий Игоревич,
Юров Максим Алексеевич,
группа Р4115

**Санкт-Петербург
2024**

Содержание

Описание решаемой задачи	3
Теоретическая база	3
Описание разработанной системы	3
Результаты работы и тестирования системы	5
Зависимость производительности при изменении размера входных данных (размера изображения)	7
Влияние ручной векторизации алгоритма на производительность	9
Зависимость производительности от уровня оптимизации	10
Вывод	12
Источники	13

Описание решаемой задачи

Цель работы:

Изучить основы оптимизации и векторизации алгоритмов компьютерного зрения на базе процессорной системы ARM Cortex A57 MPCore + NEON.

Задание:

1. Разработать программу на C++, реализующую задание в соответствии с вариантом двумя способами: без использования векторных инструкций и с ними.
2. Оценить следующие характеристики:
 - 2.1. Зависимость производительности при изменении размера входных данных (размера изображения).
 - 2.2. Зависимость производительности от уровня оптимизации (флаги -O0, -O1, -O2, -O3) для варианта без векторных инструкций.
 - 2.3. Влияние ручной векторизации алгоритма на производительность.

Теоретическая база

Теоретическая основа проекта включает принципы параллельной обработки данных с использованием SIMD (Single Instruction, Multiple Data) и технологии NEON, которая представляет собой расширение для ARM процессоров. SIMD позволяет выполнять одну и ту же операцию одновременно на множестве данных, что значительно увеличивает производительность. Использование этой технологии особенно эффективно в задачах обработки изображений, где операции применяются к большому количеству пикселей одновременно. Помимо этого, в проекте применяются методы оптимизации производительности, такие как векторизация вычислений и эффективное управление памятью.

Описание разработанной системы

Разработанная система предназначена для выполнения операций сложения и вычитания изображений с использованием стандартных и оптимизированных алгоритмов. Принцип работы заключается в загрузке двух изображений, выполнении операций над пиксельными данными и сохранении

результата. Оптимизация достигается за счёт использования NEON-intrinsic функций для параллельной обработки пикселей, что значительно снижает время выполнения операций. Архитектура системы основана на модульной структуре: базовые функции выполняются без оптимизаций, а для повышения производительности применяются векторные операции с помощью NEON.

Ниже представлен код операций сложения и вычитания изображений без оптимизации и с использованием NEON-intrinsic функций:

```
void add_images(const uint8_t* img1, const uint8_t* img2, uint8_t*
result, int num_pixels) {
    for (int i = 0; i < num_pixels * 3; i++) {
        int sum = img1[i] + img2[i];
        result[i] = (sum > 255) ? 255 : sum; // Cap at 255
    }
}

void subtract_images(const uint8_t* img1, const uint8_t* img2,
uint8_t* result, int num_pixels) {
    for (int i = 0; i < num_pixels * 3; i++) {
        int diff = img1[i] - img2[i];
        result[i] = (diff < 0) ? 0 : diff; // Floor at 0
    }
}

void add_images_neon(const uint8_t* img1, const uint8_t* img2,
uint8_t* result, int num_pixels) {
    num_pixels *= 3; // Account for RGB channels
    int i = 0;
    for (; i <= num_pixels - 16; i += 16) {
        uint8x16_t a = vld1q_u8(img1 + i);
        uint8x16_t b = vld1q_u8(img2 + i);
        uint8x16_t sum = vqaddq_u8(a, b); // Saturating addition
(caps at 255)
        vst1q_u8(result + i, sum);
    }

    for (; i < num_pixels; i++) {
        int sum = img1[i] + img2[i];
        result[i] = (sum > 255) ? 255 : sum;
    }
}
```

```

void subtract_images_neon(const uint8_t* img1, const uint8_t* img2,
uint8_t* result, int num_pixels) {
    num_pixels *= 3;
    int i = 0;
    for (; i <= num_pixels - 16; i += 16) {
        uint8x16_t a = vld1q_u8(img1 + i);
        uint8x16_t b = vld1q_u8(img2 + i);
        uint8x16_t diff = vqsubq_u8(a, b); // Saturating subtraction
        (floors at 0)
        vst1q_u8(result + i, diff);
    }

    for (; i < num_pixels; i++) {
        int diff = img1[i] - img2[i];
        result[i] = (diff < 0) ? 0 : diff;
    }
}

```

Результаты работы и тестирования системы

Для тестирования системы был использован набор изображений и их grayscale-вариаций разных разрешений, от 256 x 256 до 6000 x 4000 (рис.1).



















	Landscape_Small.jpg	17.10.2024 15:27	Файл "JPG"	18 КБ	256 x 256
	Landscape_Small_g...	15.10.2024 15:08	Файл "PNG"	36 КБ	256 x 256
	Lenna.png	24.09.2024 15:20	Файл "PNG"	463 КБ	512 x 512
	Lenna_gs.png	24.09.2024 15:20	Файл "PNG"	295 КБ	512 x 512
	Landscape_Normal....	17.10.2024 15:30	Файл "JPG"	393 КБ	980 x 551
	Landscape_Normal...	15.10.2024 15:06	Файл "PNG"	258 КБ	980 x 551
	Laptop_Normal.jpg	17.10.2024 15:22	Файл "JPG"	100 КБ	1050 x 700
	Laptop_Normal_gs....	15.10.2024 15:12	Файл "PNG"	216 КБ	1050 x 700
	tree_gs.png	22.09.2020 14:08	Файл "PNG"	1 234 КБ	1280 x 720
	tree.png	24.09.2024 15:20	Файл "PNG"	2 921 КБ	1280 x 720
	landscape_Big.jpg	17.10.2024 15:36	Файл "JPG"	1 124 КБ	2048 x 1365
	landscape_Big_gs.p...	15.10.2024 15:04	Файл "PNG"	1 789 КБ	2048 x 1365
	Laptop_MorethanN...	17.10.2024 15:24	Файл "JPG"	1 019 КБ	3800 x 2534
	Laptop_MorethanN...	15.10.2024 15:11	Файл "PNG"	2 928 КБ	3800 x 2534
	Laptop_Big.jpg	17.10.2024 15:23	Файл "JPG"	2 446 КБ	5760 x 3840
	Laptop_Big_gs.png	15.10.2024 15:10	Файл "PNG"	7 664 КБ	5760 x 3840
	Landscape_Huge.jpg	17.10.2024 15:35	Файл "JPG"	2 138 КБ	6000 x 4000
	Landscape_Huge_g...	15.10.2024 15:05	Файл "PNG"	11 752 КБ	6000 x 4000

Рис.1. Список использованных изображений.

Часть файлов, представленных выше, является вариацией одного и того же изображения, представленного в различных разрешениях (Landscape_Small.jpg, Landscape_Big.jpg и т.д.).

В результате выполнения программы базовым алгоритмом и вариантом с векторными инструкциями, были получены следующие изображения, являющиеся сложением и вычитанием исходных изображений с их grayscale-версиями:



Рис.2. Исходное изображение Landscape_Small, grayscale-версия изображения, результат их сложения и результат их вычитания (слева направо).



Рис.3. Исходное изображение Lenna, grayscale-версия изображения, результат их сложения и результат их вычитания.



Рис.4. Исходное изображение Laptop_Normal, grayscale-версия изображения, результат их сложения и результат их вычитания.

Зависимость производительности при изменении размера входных данных (размера изображения)

При определении зависимости производительности при изменении размера входных данных (размера изображения), были получены следующие результаты:

```
Processing ./img/landscape_Big.jpg with ./img/landscape_Big_gs.png...
Processing Image: landscape_Big
Image Size: 2048 x 1365
Basic addition took: 181341 us
NEON addition took: 19387 us
Basic subtraction took: 180350 us
NEON subtraction took: 19585 us
Processing ./img/Landscape_Huge.jpg with ./img/Landscape_Huge_gs.png...
Processing Image: Landscape_Huge
Image Size: 6000 x 4000
Basic addition took: 1134966 us
NEON addition took: 141659 us
Basic subtraction took: 1104805 us
NEON subtraction took: 139929 us
Processing ./img/Landscape_Normal.jpg with ./img/Landscape_Normal_gs.png...
Processing Image: Landscape_Normal
Image Size: 980 x 551
Basic addition took: 27335 us
NEON addition took: 3129 us
Basic subtraction took: 27000 us
NEON subtraction took: 3161 us
Processing ./img/Landscape_Small.jpg with ./img/Landscape_Small_gs.png...
Processing Image: Landscape_Small
Image Size: 256 x 256
Basic addition took: 3151 us
NEON addition took: 426 us
Basic subtraction took: 3045 us
NEON subtraction took: 392 us
Processing ./img/Laptop_Big.jpg with ./img/Laptop_Big_gs.png...
Processing Image: Laptop_Big
Image Size: 5760 x 3840
Basic addition took: 1035031 us
NEON addition took: 135294 us
Basic subtraction took: 1016830 us
NEON subtraction took: 134857 us
Processing ./img/Laptop_MorethanNormal.jpg with
./img/Laptop_MorethanNormal_gs.png...
Processing Image: Laptop_MorethanNormal
Image Size: 3800 x 2534
Basic addition took: 460085 us
NEON addition took: 62096 us
Basic subtraction took: 451582 us
NEON subtraction took: 62615 us
Processing ./img/Laptop_Normal.jpg with ./img/Laptop_Normal_gs.png...
Processing Image: Laptop_Normal
Image Size: 1050 x 700
Basic addition took: 35720 us
NEON addition took: 4274 us
Basic subtraction took: 33968 us
NEON subtraction took: 4233 us
Processing ./img/Lenna.png with ./img/Lenna_gs.png...
```

```

Processing Image: Lenna
Image Size: 512 x 512
Basic addition took: 12726 us
NEON addition took: 1540 us
Basic subtraction took: 12369 us
NEON subtraction took: 1554 us
Processing ./img/tree.png with ./img/tree_gs.png...
Processing Image: tree
Image Size: 1280 x 720
Basic addition took: 73888 us
NEON addition took: 8977 us
Basic subtraction took: 68196 us
NEON subtraction took: 10124 us

```

На рисунке 5 представлен график зависимости времени в микросекундах выполнения сложения и вычитания изображения с использованием базового алгоритма и варианта с векторными инструкциями от размера входных данных (также на рис.6 представлен график для первых 5 значений):

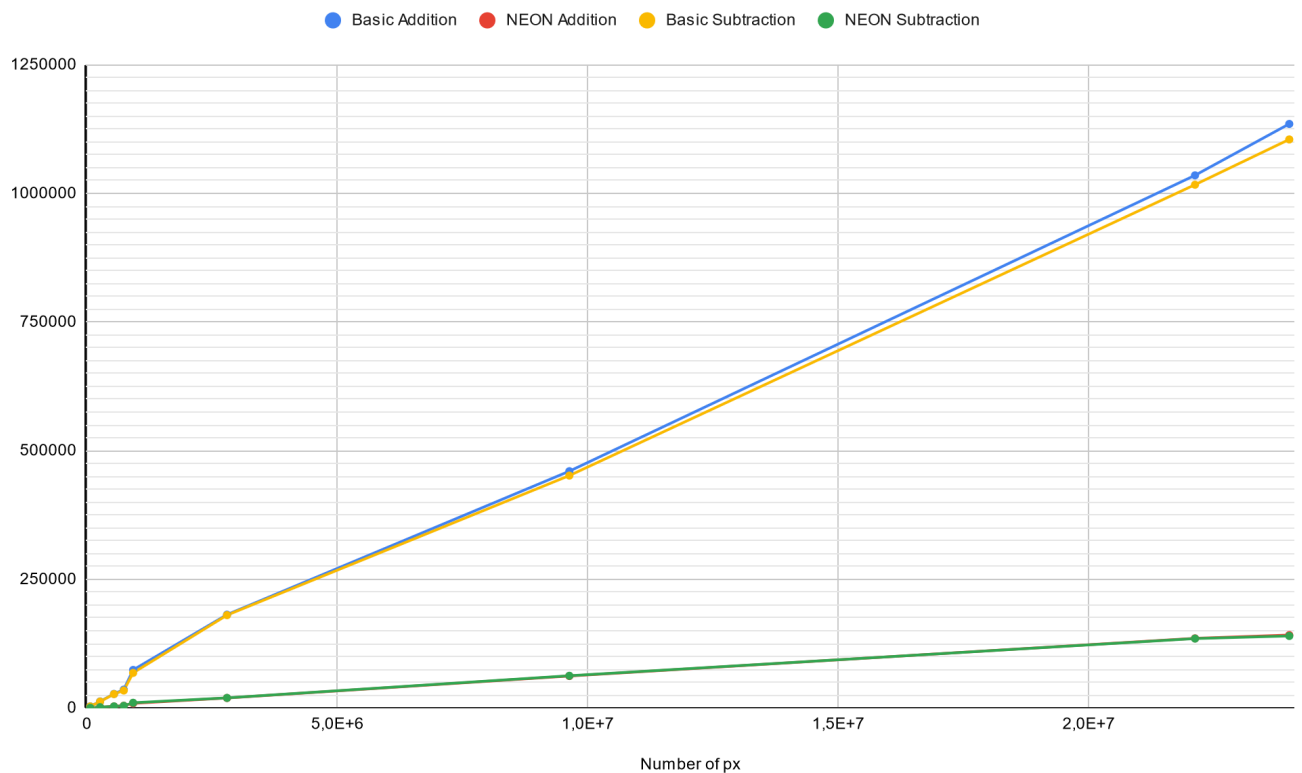


Рис. 5. График зависимости времени выполнения сложения и вычитания изображения в микросекундах от размера входных данных.

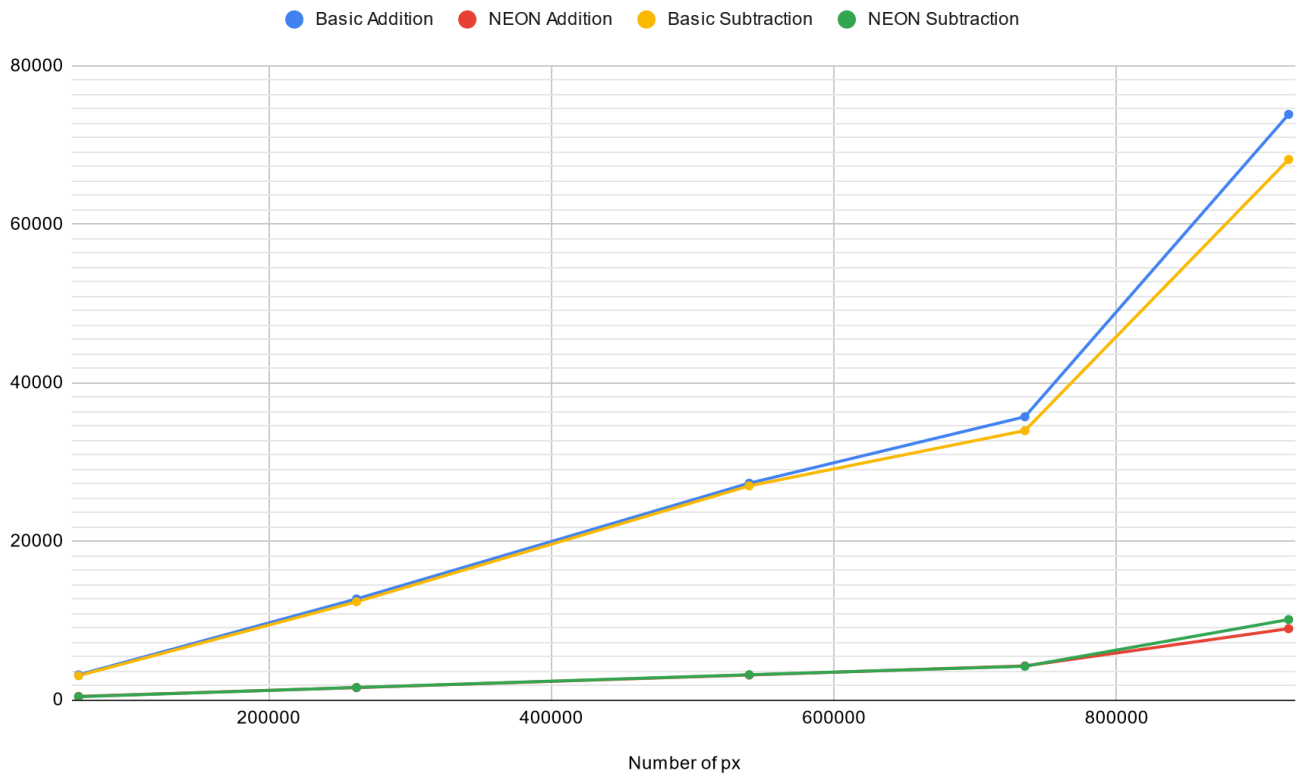


Рис. 6. График зависимости времени выполнения сложения и вычитания изображения в микросекундах от размера входных данных (для 5 начальных значений).

На основании полученных данных можно сделать вывод, что производительность программы зависит от размера изображения. С увеличением разрешения изображения (размера входных данных) время выполнения операций сложения и вычитания возрастает. Например, для небольших изображений, таких как 256x256, базовая операция сложения занимает всего около 3151 микросекунды, тогда как для больших изображений, таких как 6000x4000, время возрастает до 1 134 966 микросекунд. В то же время оптимизированные функции с использованием NEON-инструкций показывают значительное улучшение производительности на всех размерах изображений, хотя рост времени также пропорционален размеру изображения.

Влияние ручной векторизации алгоритма на производительность

Ручная векторизация с использованием NEON-инструкций приводит к значительному увеличению производительности по сравнению с базовым алгоритмом. Например, для изображения размером 6000x4000 пикселей базовая операция сложения выполняется за 1 134 966 микросекунд, тогда как оптимизированная версия с NEON — за 141 659 микросекунд, что примерно в 8 раз быстрее. Даже для небольших изображений, таких как 512x512, ускорение

заметно: NEON-сложение занимает 1540 микросекунд при 12 726 микросекунд для базового метода. Это демонстрирует, что SIMD-инструкции значительно сокращают время выполнения, особенно на больших объемах данных, за счет параллельной обработки сразу нескольких элементов данных.

Таким образом, использование NEON-инструкций дает значительное преимущество в производительности, особенно на больших изображениях, где разница между базовым и векторизованным алгоритмами наиболее заметна.

Зависимость производительности от уровня оптимизации

При определении зависимости производительности от уровня оптимизации (флаги -O0, -O1, -O2, -O3), были получены следующие результаты (в качестве тестовых изображений были использованы Lenna.png, tree.png, и Laptop_Big.jpg, размерами 512 x 512, 1280 x 720 и 5760 x 3840 соответственно):

```
Running image_processing_-O0 on Lenna.png and ./img/Lenna_gs.png...
Processing Image: Lenna
Image Size: 512 x 512
Basic addition took: 13573 us
NEON addition took: 1594 us
Basic subtraction took: 12701 us
NEON subtraction took: 1678 us
Running image_processing_-O0 on tree.png and ./img/tree_gs.png...
Processing Image: tree
Image Size: 1280 x 720
Basic addition took: 46433 us
NEON addition took: 6214 us
Basic subtraction took: 43763 us
NEON subtraction took: 5766 us
Running image_processing_-O0 on Laptop_Big.jpg and
./img/Laptop_Big_gs.png...
Processing Image: Laptop_Big
Image Size: 5760 x 3840
Basic addition took: 1033761 us
NEON addition took: 135858 us
Basic subtraction took: 1019484 us
NEON subtraction took: 137465 us
Running image_processing_-O1 on Lenna.png and ./img/Lenna_gs.png...
Processing Image: Lenna
Image Size: 512 x 512
Basic addition took: 2827 us
NEON addition took: 379 us
Basic subtraction took: 1530 us
NEON subtraction took: 393 us
Running image_processing_-O1 on tree.png and ./img/tree_gs.png...
Processing Image: tree
Image Size: 1280 x 720
Basic addition took: 10701 us
NEON addition took: 1609 us
Basic subtraction took: 7198 us
NEON subtraction took: 1789 us
Running image_processing_-O1 on Laptop_Big.jpg and
./img/Laptop_Big_gs.png...
Processing Image: Laptop_Big
```

```

Image Size: 5760 x 3840
Basic addition took: 170776 us
NEON addition took: 31211 us
Basic subtraction took: 129695 us
NEON subtraction took: 31936 us
Running image_processing_-02 on Lenna.png and ./img/Lenna_gs.png...
Processing Image: Lenna
Image Size: 512 x 512
Basic addition took: 5200 us
NEON addition took: 420 us
Basic subtraction took: 4378 us
NEON subtraction took: 386 us
Running image_processing_-02 on tree.png and ./img/tree_gs.png...
Processing Image: tree
Image Size: 1280 x 720
Basic addition took: 18100 us
NEON addition took: 1244 us
Basic subtraction took: 15640 us
NEON subtraction took: 1239 us
Running image_processing_-02 on Laptop_Big.jpg and
./img/Laptop_Big_gs.png...
Processing Image: Laptop_Big
Image Size: 5760 x 3840
Basic addition took: 383079 us
NEON addition took: 33625 us
Basic subtraction took: 365217 us
NEON subtraction took: 38104 us
Running image_processing_-03 on Lenna.png and ./img/Lenna_gs.png...
Processing Image: Lenna
Image Size: 512 x 512
Basic addition took: 5022 us
NEON addition took: 385 us
Basic subtraction took: 4364 us
NEON subtraction took: 435 us
Running image_processing_-03 on tree.png and ./img/tree_gs.png...
Processing Image: tree
Image Size: 1280 x 720
Basic addition took: 17633 us
NEON addition took: 1513 us
Basic subtraction took: 15402 us
NEON subtraction took: 1441 us
Running image_processing_-03 on Laptop_Big.jpg and
./img/Laptop_Big_gs.png...
Processing Image: Laptop_Big
Image Size: 5760 x 3840
Basic addition took: 383303 us
NEON addition took: 33423 us
Basic subtraction took: 368549 us
NEON subtraction took: 35584 us

```

На рис. 7 представлен график зависимости времени выполнения сложения изображений базовым алгоритмом в микросекундах от уровня оптимизации (флаги -O0, -O1, -O2, -O3) на тестовых изображениях.

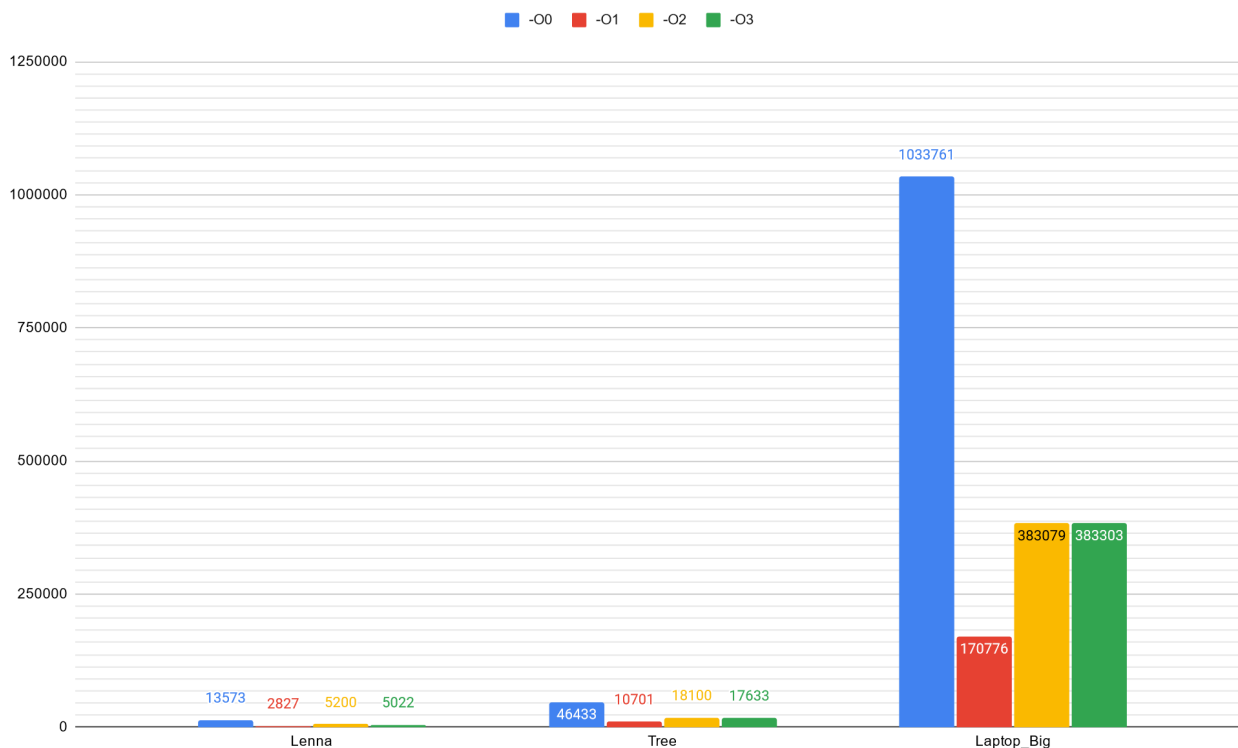


Рис. 7 график зависимости времени выполнения сложения изображений базовым алгоритмом в микросекундах от уровня оптимизации.

Несмотря на большой скачок в производительности при переходе от уровня оптимизации O0 к O1, при установке флагов O2 и O3 время выполнения сложения выражений алгоритмом без векторных инструкций возрастает по сравнению с O1, что может быть связано с увеличением размера скомпилированного кода (например, за счет агрессивного разворачивания циклов или более сложного управления регистрами), что иногда может негативно повлиять на эффективность кэширования процессора, особенно при работе с небольшими или средними изображениями.

Вывод

В результате выполнения работы мы ознакомились с основами оптимизации и векторизации алгоритмов компьютерного зрения на базе процессорной системы ARM Cortex A57 MPCore + NEON с использованием модуля Jetson Nano. Нами были изучены основы работы с векторными инструкциями SIMD расширения Neon, использованные для написания алгоритмов сложения и вычитания изображений.

Источники

<https://algorithmica.org/ru/sse>

<https://documentation-service.arm.com/static/63299276e68c6809a6b41308>

<https://github.com/thenifty/neon-guide#neon-intrinsics-guide>