Санкт–Петербургский государственный университет

Криворучко Денис Игоревич

Отчёт по домашней работе 1
Приложение для работы с изображениями

Санкт-Петербург
2021 г.

Содержание

Глава 1. Введение

В процессе выполнения домашней работы 1, было разработано приложение для обработки изображений. Поддержаны следующие возможности: просмотр информации об изображении (размер, высота, ширина), применение одного или нескольких матричных фильтров, выбор из заранее заданных фильтров, а также конструирование собственного. Приложение позволяет работать с одним изображением или с директорией. В случае работы с одним изображением, предусмотрена также возможность просмотреть результат наложения фильтров и, при желании, сохранить его. Если же происходит обработка списка изображений, которые хранятся в папке, полученные изображения сразу сохраняются. Важной особенностью реализации является возможность применения фильтров с помощью центрального процессора (CPU) или видеокарты (GPU).

Данный отчет содержит обзор разработанного продукта, а также анализ производительности и сравнение скорости обработки на CPU и на GPU.

Глава 2. Используемые инструменты

При разработке приложения использован язык Python 3. Для Frontend-части выбран фреймворк PyQt5 [1], который предоставляет большое количество прмитивов для взаимодействия с пользователем. Поскольку большая часть работы в backend-части программы связана с обработкой матрицы изображений, при расчетах на CPU, используется модуль NumPy [2]. Данный модуль подходит для задач, связанных с линейной алгеброй, так как здесь базовые операции над матрицами и векторами отлично оптимизированы. Для вычисления на видекарте использована библиотека CuPy [3]. CuPy является аналогом NumPy, за исключением того, что расчеты проводятся на GPU. На самом деле, CuPy — это оболочка библиотеки CUDA [4] языка C, разработанной для взаимодействия с видекартами NVIDIA.

Глава 3. Детали реализации

3.1 Frontend

Приложение содержит 6 окон:

1. main_view — для выбора режима работы (обработка одного изображения или директории)
2. one_picture_view — для просмтора инофрмации об изображении
3. filters_view — для выбора фильтров, которые будут применены к изображению
4. create_filter_view — для создания фильтра
5. two_pictures_view — для сравнения изображения до и после применения выбранных фильтров
6. saved_view — для просмотра папки, в которую сохранен результат наложения фильтров

Все взаимодействие с frontend- и backend-частей происходит через filters_view.

3.2 Backend

Backend-часть приложения состоит из трех основных компонентов. Первый из них — computing, отвечает за взаимодействие с frontend. Модули cpu_computing и gpu_computing нужны для наложения фильтров с использованием CPU и GPU соответственно.

Наложение матричных фильтров происходит следующим образом: фильтр задается некоторой матрицей F с нечетными высотой h и шириной w. Далее фильтр применяется к каждому пикселю матрицы изображения следующим образом: в изображении выбирается подматрица M, размерности которой совпадают с размерностями матрицы фильтра. Причем текущий пиксель должен являться центральным в M. После этого формируется новая матрица: она представляет из себя поэлементное умножение

матрицы F на матрицу M. Значение пикселя в изоражении, которое является резльтатом применения фильтра, — это сумма элементов матрицы M. Для того, чтобы работать с пикселями, которые находятся на границе изображения, формируется расширенная матрица изображения. Если высота матрицы фильтра равна h, а высота матрицы изображения равна h', то высота расширенной матрицы: $H = h + h' - 1$. Ширина расширенной матрицы определяется аналогично.

В реализации модуля cpu_computing все действия происходят последовательно. При применении фильтра, сначала формируется расширенная матрица изображения. Затем фильтр применяется отдельно к каждому пикселю. Если фильтров и изображений несколько, то каждый фильтр применяется поочередно к каждому изображению.

В реализации модуля gpu_computing применение фильтра к изображению происходит параллельно, с использованием нескольких ядер графического процессора. Это реализовано с помощью возможностей библиотеки Cupy и дает существенный рост производительности даже с учетом того, что все данные о фильтрах и изображениях необходимо сначала предать видеокарте, а после обработки, вернуть информацию на центральный процессор. Случай применения нескольких фильтров к нескольким изображениям реализован так же, как и в cpu_computing: каждый фильтр применяется поочередно к каждому изображению.

Благодаря тому, что frontend и backend полученного решения связаны единственным методом, приложение является расширяемым. Например, для того, чтобы добавить новый стандартный фильтр, необходимо создать одну кнопку в filters_view, и указать матрицу фильтра в кофигурационном файле. Если же потребуется новый способ применения фильтров (например на CPU, но параллельно), backend таких вычислений нужно реализовать аналогично с двумя другими, уже существующими модулями, а в окне filters_view добавить одну кнопку.

Глава 4. Постановка экспериментов и измерение производительности

4.1 Эксперименты

Измерения производились на компьтере с процессором INTEL Core i5 8th Gen (2.30GHz), видеокартой NVIDIA GeForce GTX 1050ti, 16 гб оперативной памяти и операционной системой Ubuntu 20.04.

При постановке экспериментов вычислялось только время наложения фильтров. В случае рассчетов на GPU, учитывалось также время передачи данных между видеокартой и центральным процессором. Каждый эксперимент проводился 10 раз. Всякий раз фильтры генерировались случайно. В качестве результатов проведения каждого эксперимента выбрано среднее время работы по 10 замерам.

Первый эксперимент заключался в следующем: производительность решений с использованием CPU и GPU сравнивалась отдельно на трех изображениях: 246×205, 275×183, 646×1000. К каждому из них применялось два фильтра: 5×5 и 3×3. Результаты проведения эксперимента представлены на рисунке 1.

Второй эксперимент был таким: к 5 изображениям из выбранной директории, которая содержала изображения следующих размеров: 246×205, 275×183, 646×1000, 1600×1200, 220×165, применялось 5 фильтров 5×5. Результаты проведения эксперимента представлены на рисунке 3.

В третьем эксперименте вычислялось время применения фильтра 1001×1001 к изображению 646×1000. Результаты представлены на рисунке **??**.

4.2 Анализ

Несмотря на то, что при использовании видеокарты в расчетах, время уходит не только на применение фильтров, но и на обмен данными между графическим и центральным процессорами, реализация на GPU показывает значительно лучшие результаты. В среднем, при работе с CPU, время, затрачиваемое на обработку одного изображения, более, чем в 300 раз превосходит время, которое требуется при наложении фильтров на GPU.

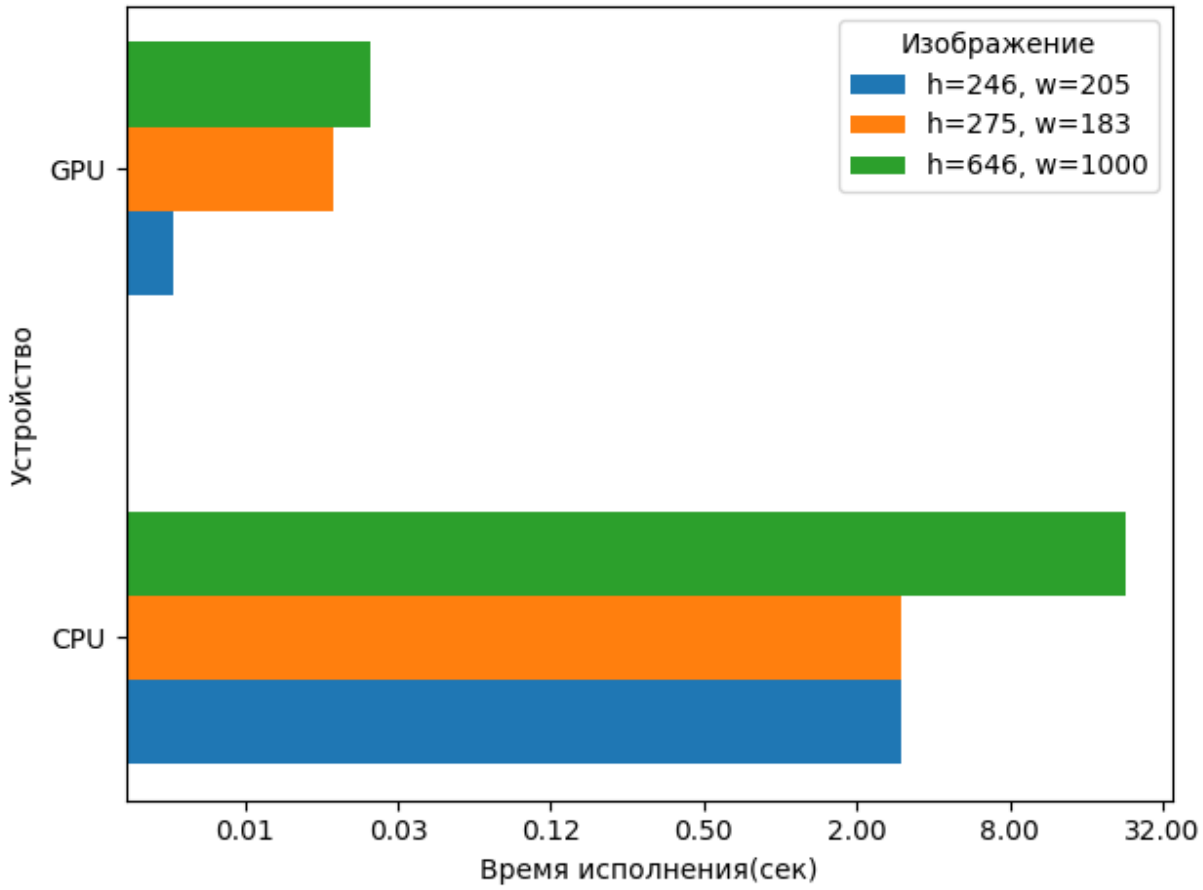

Рис. 1: Применение фильтров 5×5 и 3×3 к изображению.

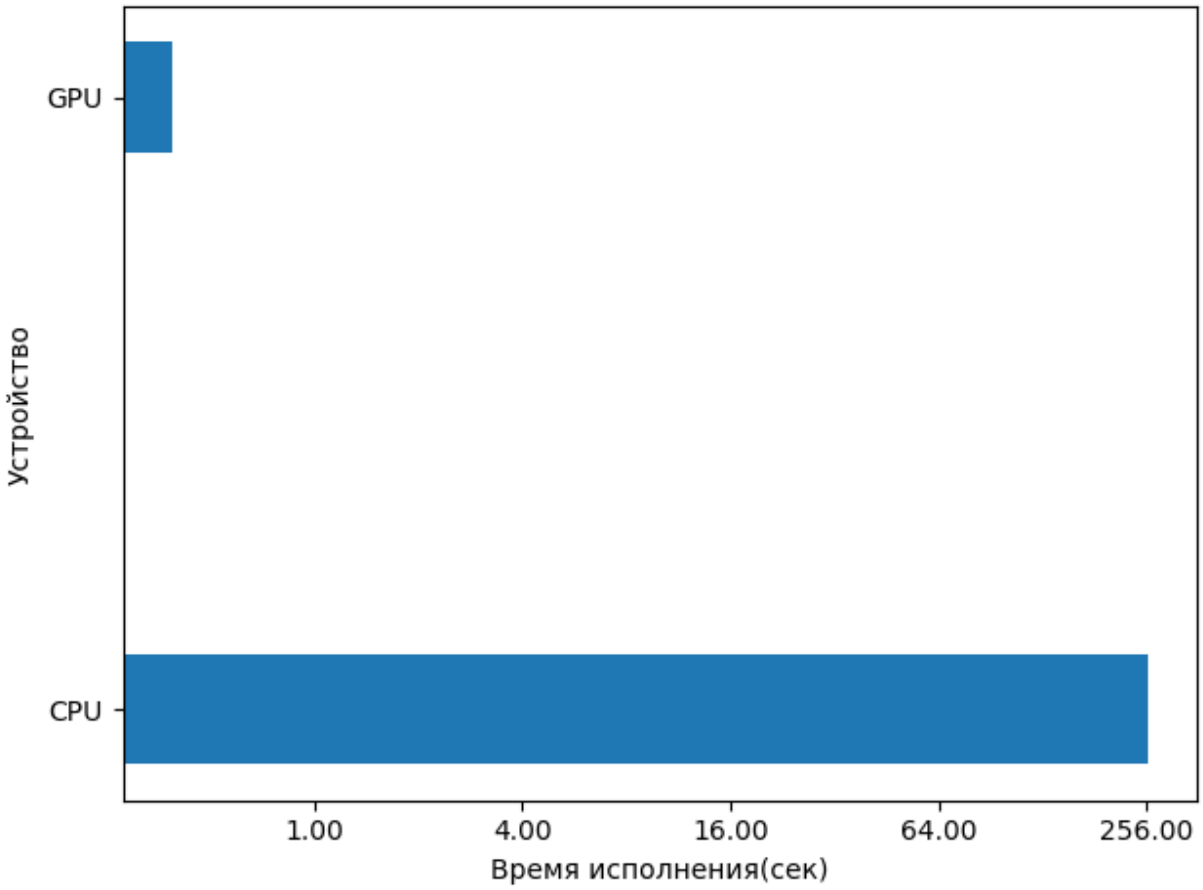

Рис. 2: Применение 5 фильтров 5×5 к папке из 5 изображений.

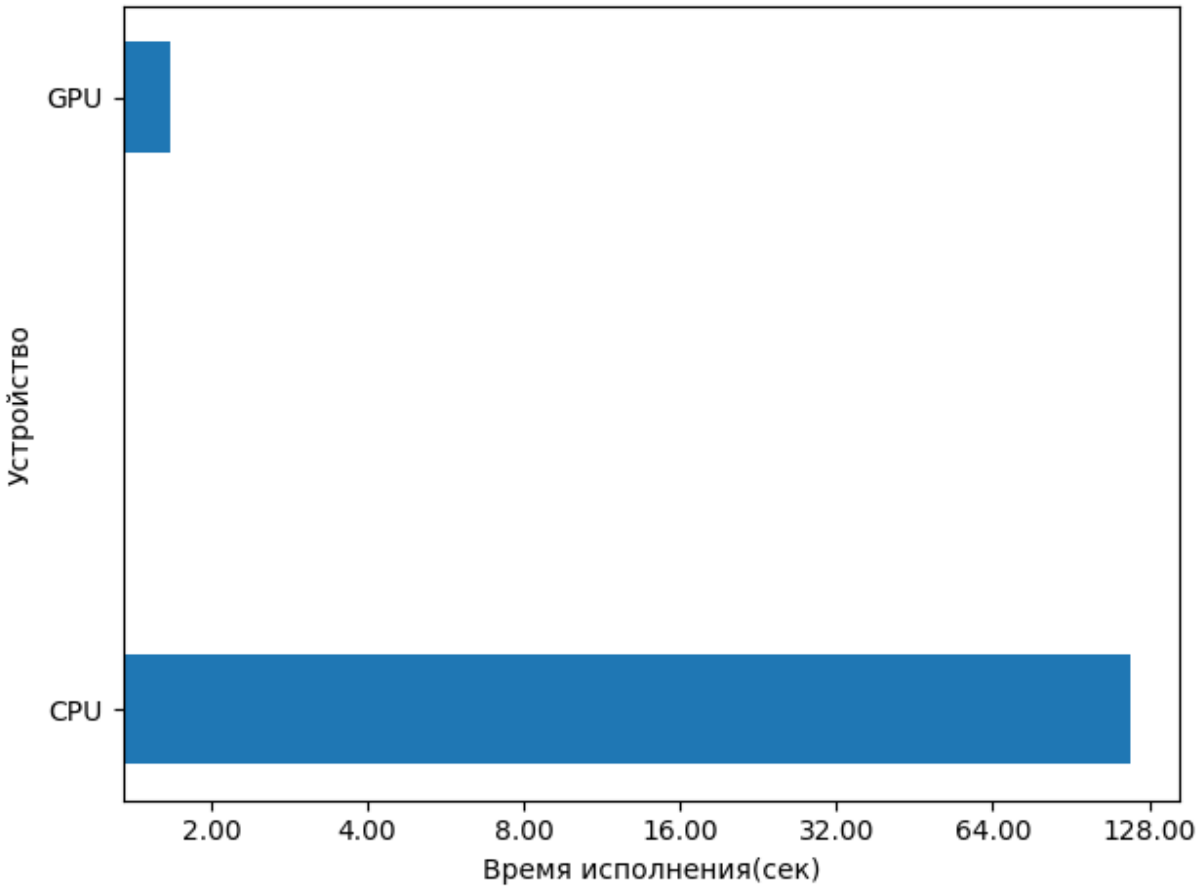

Рис. 3: Применение фильтра 101×101 к изображению 646×1000.

Существенно, что при увеличении входных данных, время рассчетов на графическом процессоре увеличивается незначительно — менее, чем в 2 раза, при увеличении изображения примерно в 12 раз, в то время, как время обработки на CPU при таком же увеличении данных, растет более, чем в 7 раз.

Что касается работы с несколькими изображениями, на GPU время растет незначительно и при применении 5 фильтров к 5 изображениям остается меньшим секунды. Однако, те же расчеты на CPU занимают более 4 минут.

При увеличении матрицы фильтра до размеров 101×101, разница во времени обработки одного изображения на CPU и GPU становится не такой значительной. Вероятно, это связано с транспортировкой данных между графическим и центральным процессорами. Однако, рассчеты на видеокарте все еще требуют почти в 100 раз меньше времени.

Глава 5. Заключение

На основе проведенных экспериментов можно сделать вывод, что для решения задач линейной алгебры, в которых требуется со всеми элементами матриц или векторов производить однообразные независимые операции, например, при наложении матричных фильтров на изображение, использование GPU является более оптимальным, чем использование CPU.

Список литературы

[1] PyQt documentation. URL: https://doc.qt.io/qtforpython/

[2] NumPy documentation. URL: https://numpy.org/

[3] CuPy documentation. URL: https://docs.cupy.dev/en/stable/

[4] CUDA documentation. URL: https://docs.nvidia.com/cuda/