Оптимизация алгоритма SGM для архитектур на базе GPGPU

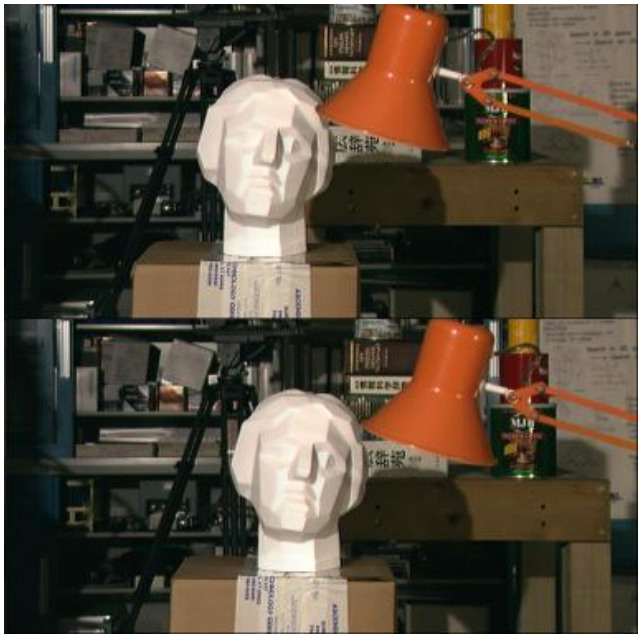
Выполнил студент 545 группы

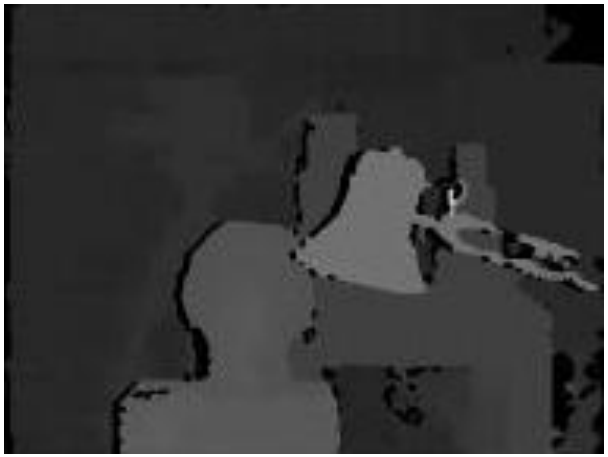
Шевченко Андрей

Научный руководитель – Пименов А. А.

Рецензент – Кривошеин Б. Н.

Введение в проблему





Введение в проблему

- Диспаратность – относительная глубина пикселя изображения
- Построение карты диспаратностей – сложная вычислительная задача
- Проект DeepView – разработка ускорителя для распознавания

Обзор существующих решений

- Локальные алгоритмы
  - Вычисление диспаратности происходит для каждого пикселя в отдельности
  - Преимущества – высокая скорость работы
  - Проблемы – некачественные результаты с большим числом ошибок
- Глобальные алгоритмы
  - Поиск карты диспаратности происходит для всего изображения сразу
  - Преимущества – очень хорошие результаты с малым числом ошибок, учитываются связи между пикселями всего изображения
  - Проблемы – низкая скорость работы, не предназначены для задач реального времени

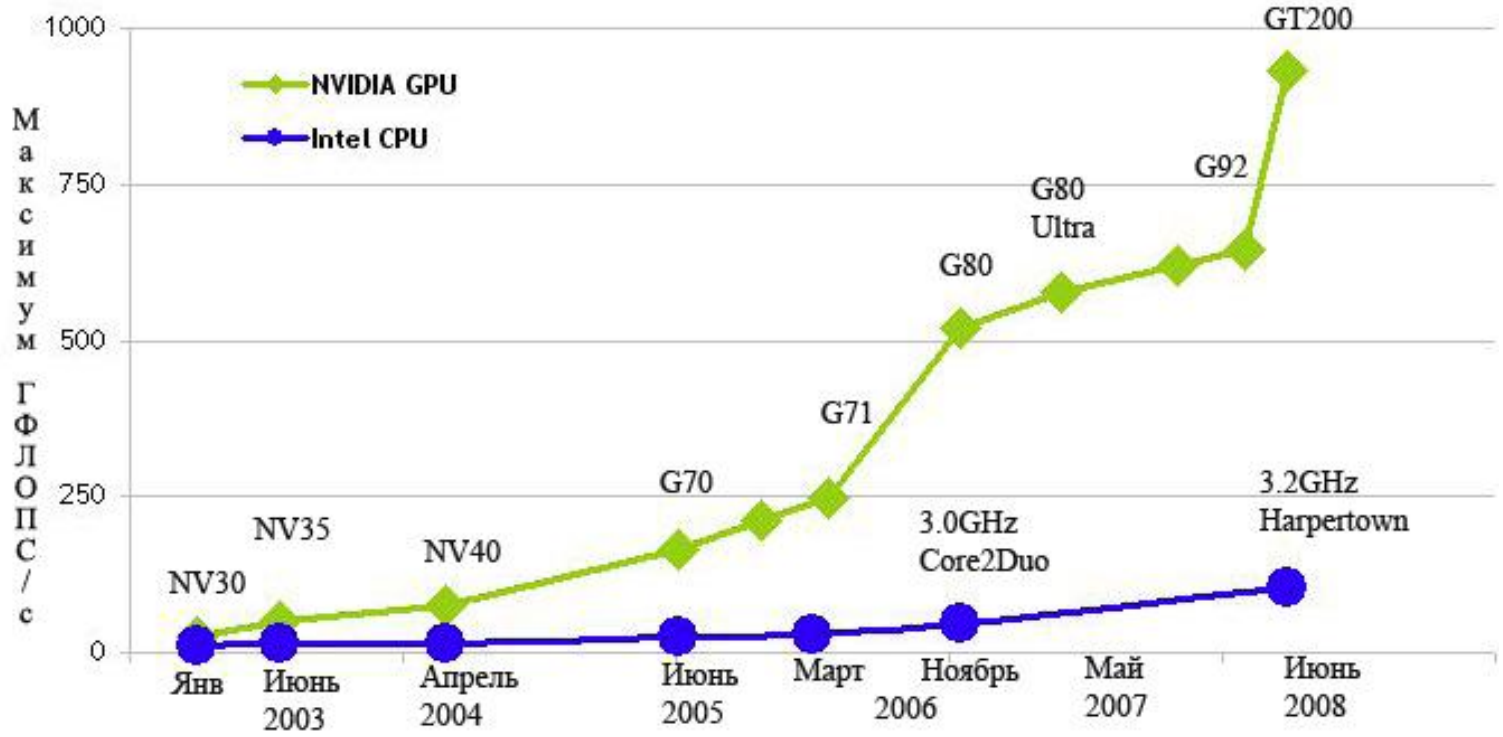
Алгоритм SGM

- SGM – Semi-Global Matching
- Алгоритм был разработан в 2005 году немецким профессором Heiko Hirschmuller
- Шаги алгоритма
  - Итеративный подсчет попиксельных стоимостей на основе взаимной информации
  - Суммирование стоимостей по нескольким направлениям
  - Нахождение диспаратности для каждого пикселя

CUDA

- Графический процессор – высокопараллельное устройство с очень большой вычислительной мощностью
- GPGPU – general purpose computing on graphic processor units – вычисления общего назначения на графическом процессоре
- CUDA - технология GPGPU , позволяющая реализовывать алгоритмы, выполнимые на графическом процессоре
- Создана в 2006 году компанией NVIDIA

CUDA



GT200 = GeForce GTX 280
G92 = GeForce 9800 GTX
G80 = GeForce 8800 GTX
G71 = GeForce7900 GTX

G70 = GeForce 7800 GTX
NV40 = GeForce 6800 Ultra
NV35 = GeForce FX 5950 Ultra
NV30 = GeForce Fx 5800

Цели работы

- Анализ алгоритма SGM для переноса на массово-параллельные архитектуры
- Реализация модифицированного SGM на платформе CUDA
- Исследование и внесение предложений по оптимизации алгоритма
- Количественная и качественная оценка получившегося алгоритма

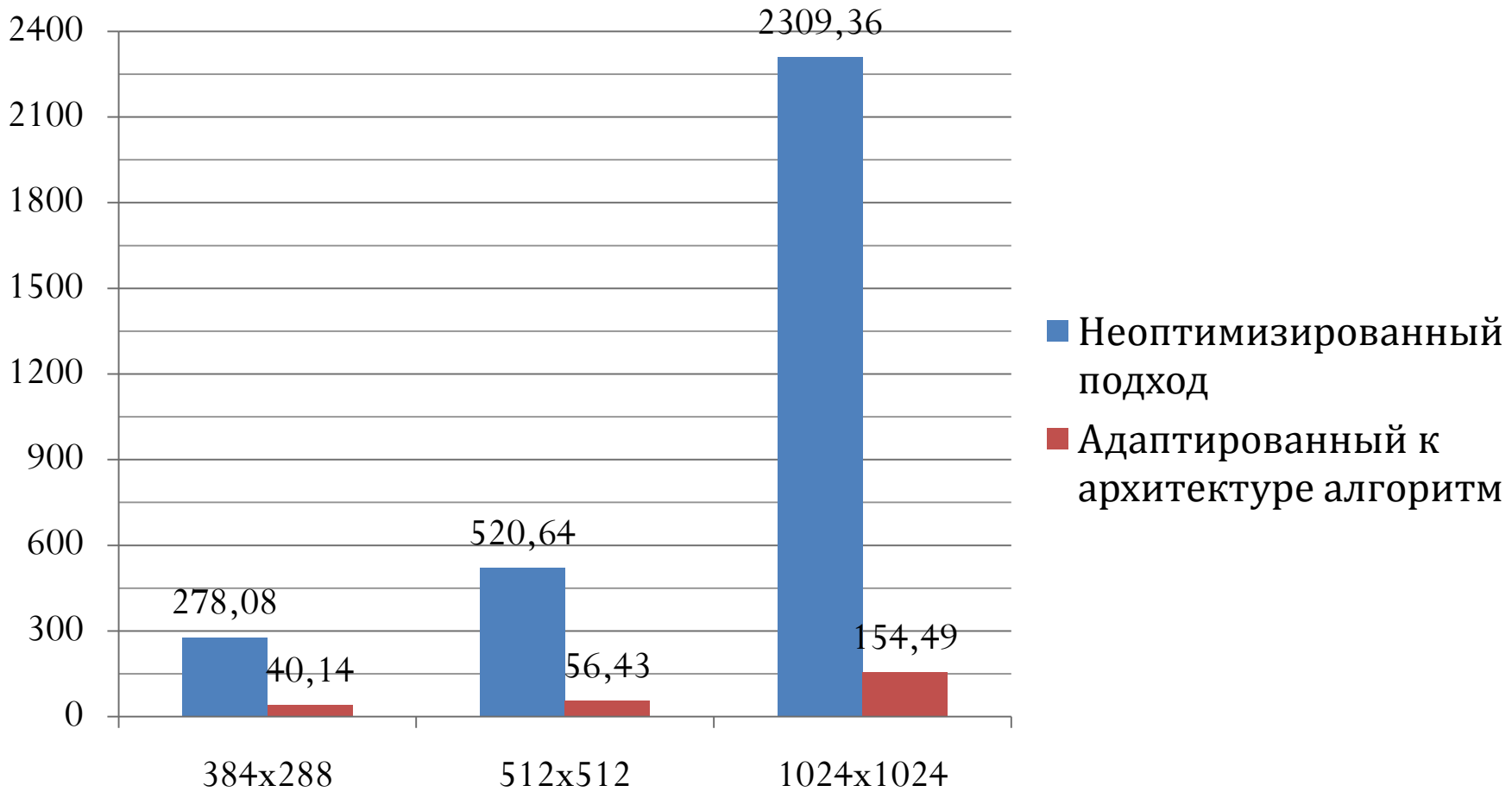
Проблемы с памятью

- Алгоритму требуются большие объемы памяти
  - Для изображения 1024× 1024 – около 400 Мб
- CUDA позволяет выделить всего 512 Мб
- Проблема решалась несколькими способами
  - Использование типа half-float вместо float
    - Низкая скорость конвертации
  - Разбиение на тайлы
    - Поддержка изображений очень больших размеров
    - Необходимо, чтобы тайлы пересекались => скорость работы снижается
- Второй способ лучше!

Вычисление взаимной информации

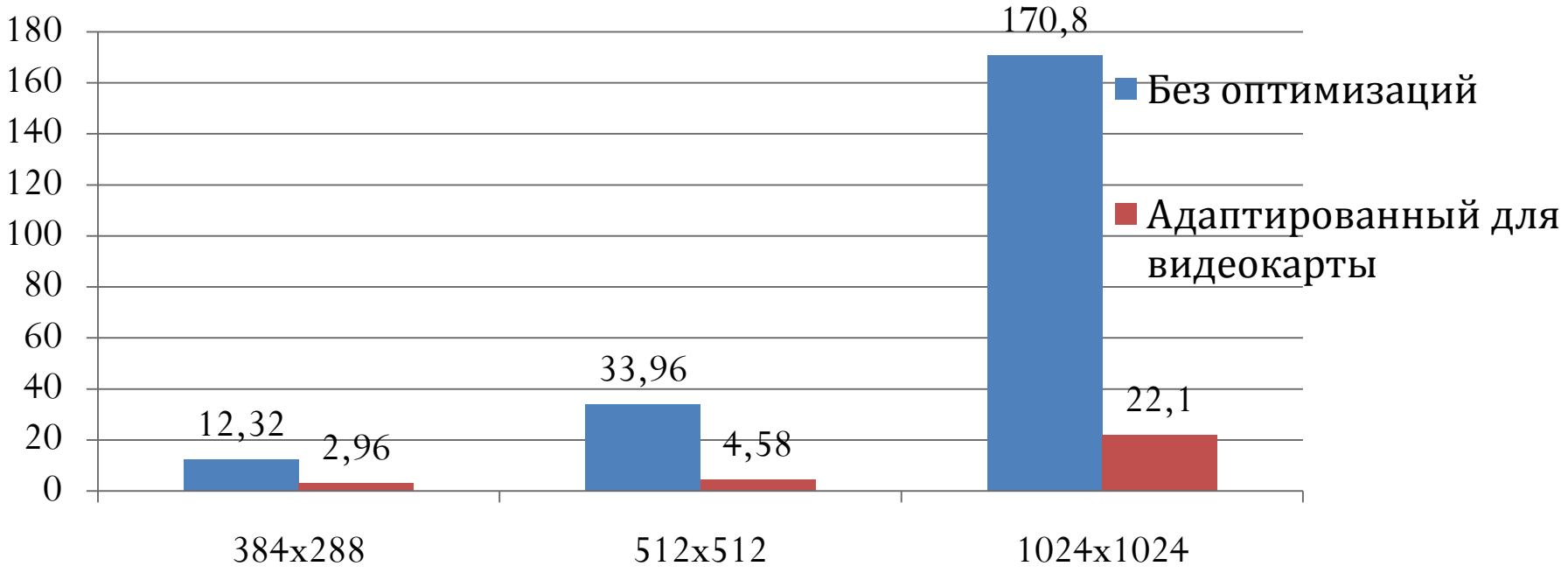
- Взаимная информация – количество информации, содержащееся в одной случайной величине относительно другой
- Используется для подсчета попиксельных стоимостей
  - Преимущества – вычисленные стоимости будут корректны, даже если пара изображений снята при абсолютно разных настройках камер
  - Недостатки – скорость работы ниже, чем при использовании более простых подходов

Вычисление взаимной информации



Модифицированный алгоритм абсолютных разностей

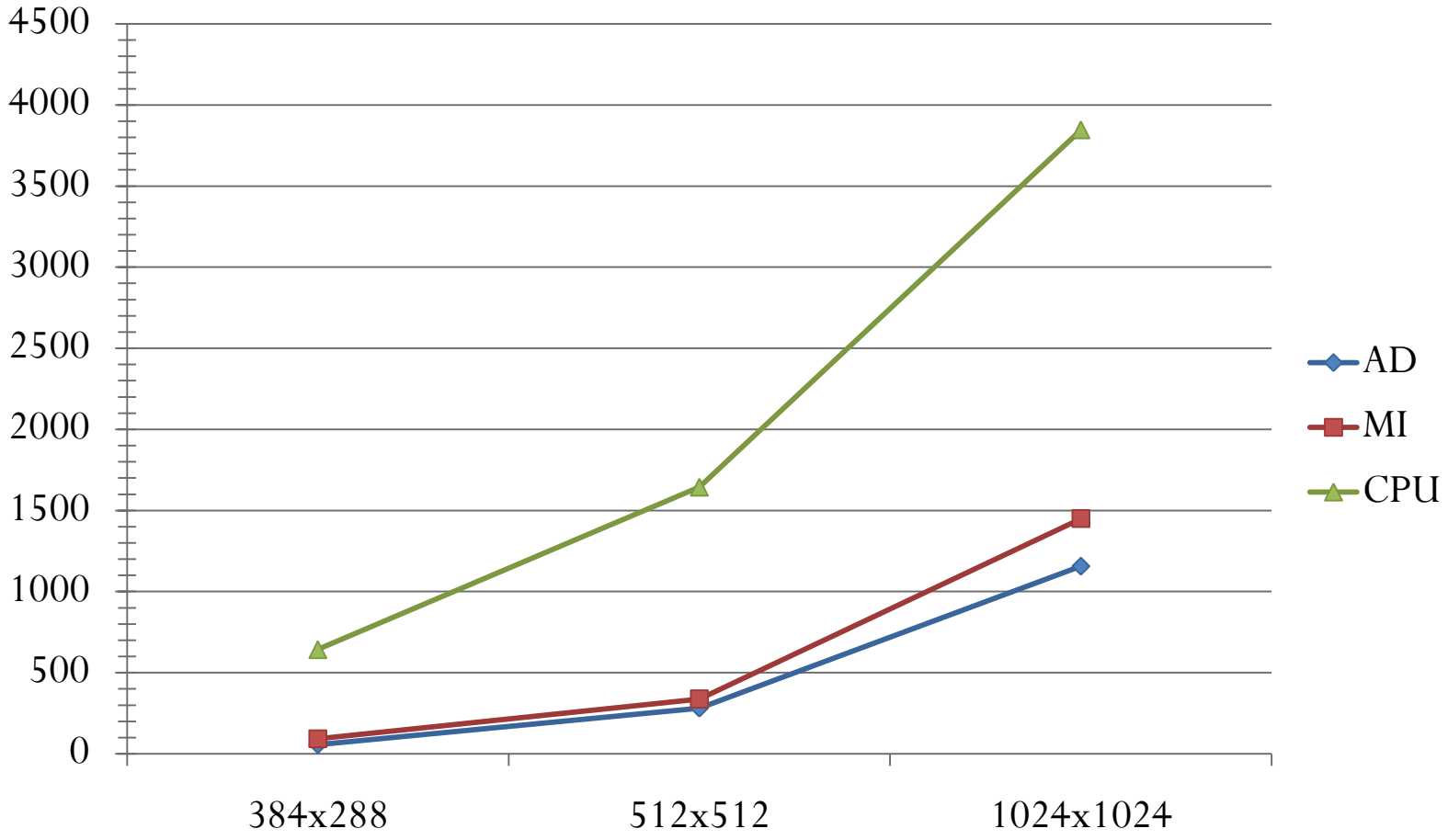
- Стоимость для данной пары пикселей – модуль разности их интенсивностей
- Скорость выше, чем у предыдущего подхода
- Процент ошибок тоже выше



Суммирование стоимостей

- Попиксельные стоимости суммируются по нескольким направлениям
  - Оригинальный алгоритм – 16 направлений
  - Модификация – используется либо 8 направлений, либо 5
- Итоговый процент ошибок составляет от 4% до 6%
- Сравнимо с глобальными подходами, где количество ошибок от 2 до 3 процентов

Время работы



Результаты

- Был разработан модифицированный алгоритм на основе SGM, предназначенный для массово-параллельных архитектур
- Полученный алгоритм был реализован на платформе CUDA
- Полностью решена проблема нехватки памяти
- Были разработаны и реализованы различные оптимизации для модифицированного алгоритма