**Обработка изображений: «Повышение четкости изображений»**

Язык программирования: С++

Алгоритм: вычитание из изображения оператора Лапласа

Описание входа и выхода: 122 изображения в формате RGB(8-bit) jpg (75 Mb).

Метод сравнения качества изображений: визуальный.

**Описание алгоритма:**

Оператор Лапласа – линейный фильтр, который представляет собой вторую производную, которая подчеркивает разрывы уровней яркости на изображении. Эффект повышения резкости достигается вычитанием лапласиана из изображения.

Ссылка на репозиторий с кодом: <https://github.com/VeronikaVasilyeva/Laplacian>

**Методы оптимизации:**

**Изменение программного кода (**скалярные оптимизации и оптимизации циклов**):**

* 1. Скалярная оптимизация: Удаление повторных вычислений в get\_pixel, set\_pixel, applay\_filter: вынесено во временную переменную tmp.
  2. Оптимизации циклов:
     1. Перестановка циклов в laplacian: Изображения хранятся в памяти в виде одномерных массивов, записываются построчно. Поэтому обрабатывать их необходимо последовательно, это быстрее, чем произвольный доступ к памяти. Во внешнем цикле обход по вертикали, во внутреннем – по горизонтали (построчно). Также это исправление полезно для автоматической векторизации.
     2. Развертка цикла в laplacian: Удален цикл перебирающий каналы. Теперь вычисляем сразу все три.
  3. Подстановка в главный поток 4х функций: get\_pixel, set\_pixel, applay\_filter, laplacian
  4. Были общие глобальные переменные: resourse, result, width, height.   
     Они стали локальными для того, чтобы было удобно распараллеливать.

Время работы: 120 сек

**Добавление ключей оптимизации: icc –O2 –ipo –openmp main.cpp –o mainOpt**

1. Автоматическая векторизация: -O2

Отчет о векторизации O2:

main.cpp(93): (col. 3) remark: LOOP WAS VECTORIZED.

stb\_image.h(4062): (col. 2) remark: PARTIAL LOOP WAS VECTORIZED.

stb\_image.h(4429): (col. 5) remark: LOOP WAS VECTORIZED.

stb\_image.h(4428): (col. 5) remark: LOOP WAS VECTORIZED.

stb\_image.h(4427): (col. 5) remark: LOOP WAS VECTORIZED.

stb\_image.h(4411): (col. 5) remark: LOOP WAS VECTORIZED.

stb\_image.h(4410): (col. 5) remark: LOOP WAS VECTORIZED.

stb\_image.h(1545): (col. 4) remark: LOOP WAS VECTORIZED.

stb\_image.h(2846): (col. 10) remark: LOOP WAS VECTORIZED.

stb\_image.h(2850): (col. 10) remark: LOOP WAS VECTORIZED.

stb\_image.h(3126): (col. 3) remark: LOOP WAS VECTORIZED.

stb\_image.h(5593): (col. 5) remark: LOOP WAS VECTORIZED.

stb\_image.h(5593): (col. 5) remark: LOOP WAS VECTORIZED.

tiny\_jpeg.h(712): (col. 3) remark: LOOP WAS VECTORIZED.

tiny\_jpeg.h(519): (col. 3) remark: LOOP WAS VECTORIZED.

stb\_image.h(3333): (col. 3) remark: LOOP WAS VECTORIZED.

1. Межпроцедурная оптимизация: -ipo

ipo: remark #11001: performing single-file optimizations

ipo: remark #11005: generating object file /tmp/ipo\_icckAe6KF.o

1. OpenMP: -openmp

Добавлено #include <omp.h> в начало программы.

Замер времени: omp\_get\_wtime(); omp\_get\_wtime();

Добавлено перед главным циклом, который перебирает изображения:

int num\_thread = omp\_get\_num\_procs();

#pragma omp parallel for

num\_threads(num\_thread)

private(img, width, height, bpp, file\_sourse, resourse, result, file\_result)

firstprivate(path, extension, extension1, filter)

schedule(guided)

Время работы: 61 сек

**Результаты:**

1. Плохой и хороший код решает задачу корректно.
2. На снимках визуально заметно увеличение качества изображения, объекты становятся более яркими, линии становятся четче. Результаты можно посмотреть в папке DataSet, каждое изображение именовано соответствующим именем программы, также приложены исходные изображения.
3. Программа, написанная на OpenCV, работает быстрее (примерно в 20 раз относительно неоптимизированной программы), качество изображений - одинаковое.
4. После всех оптимизаций программа работает быстрее на 112 сек. Неоптимизированная в среднем работала 173 сек., оптимизированная – 61 сек.