1 Запуск

Проект собирается при помощи make и запускается в формате

./Main image_path threshold

image_path – полный путь до исходного RGB изображения

threshold – порог T

Обработанное изображение сохраняется как *out.jpg*.

2 Общее

В основе реализации лежит то, что исходное изображение парсится на 3 массива по каналам RGB, к ним применяется алгоритм медианной фильтрации и из результирующих массивов собирается новое изображение.

image_channels – вспомогательный класс для работы с изображениями (конвертирует изображение в массивы RGB и наоборот).

median_filter и *median_filter* :: *blur* – основной класс и метод, решающие задачу для изображения, приватный метод *median_filter* :: *process* решает данную задачу для матриц.

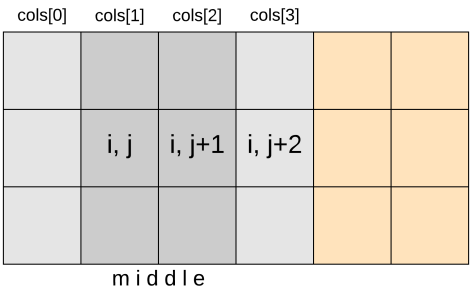
(a) input

(b) output, T = 0

Рис. 1: Пример

3 Алгоритм

Алгоритм основан на том, что у двух соседних клеток (например, $[i, j]$ и $[i, j+1]$) есть 2 общих столбца в окне 3x3, поэтому можно отсортировать эти столбцы ровно один раз и, имея отсортированную середину, при добавлении к ней $j - 1$ столбца найти медиану клетки $[i, j]$ и при добавлении $j + 2$ столбца найти медиану клетки $[i, j + 1]$).



Таким образом на одном итерации получается найти две медианы. Также можно заметить, что при переходе к клетке $[i, j + 2]$) мы уже будем иметь серые столбцы отсортированными с прошлой итерации, поэтому на каждой итерации нужно добавлять только 2 желтых столбца.

У меня реализовано так, что на каждой итерации цикла я рассматриваю окно 3x4 и храню 4 отсортированных массива *cols*, два из которых имеются с прошлой итерации, и два новых. Потом я делаю *median_filter* :: *merge* средних массивов и получаю 2 медианы при помощи метода *median_filter* :: *get_median*.

Для крайних строчек и столбцов я дополняю окно медианы нулями (для этого для крайних случаев вызывается версия $median\_filter :: process\_row$ с $safe\_get$, который возвращает 0 при некорректных индексах, иначе вызывается с обычным get без проверки индексов).

4 Количество операций

Если считать немного грубо и брать в расчет только основные операции алгоритма, то на каждой итерации цикла:

- $median\_filter :: fill\_cols$ – заполняет и сортирует массивы $cols[2]$ и $cols[3]$, т.е. $2 \cdot 3$ операций присваивания и максимум $2 \cdot 3$ операций сравнения для сортировки;
- $median\_filter :: merge$ – 5 операций сравнения элементов массивов $cols[2]$ и $cols[3]$ между собой и 4 операции присваивания;
- $median\_filter :: get\_median$ – по 4 сравнения на каждую из медиан, т.е. 8 операций сравнения;

То есть при таком подсчете при $n = 1080,\ m = 1920$ получается $1080 \cdot 1920/2 \cdot (12 + 9 + 8) = 30\,067\,200$ операций.

При более точных расчетах, учитывая некоторые детали реализации (но все равно очень приблизительно, расчеты приведены в комментариях исходного кода) у меня получилось около 93 404 772 операций.