

Медианный фильтр

Вадим Володин



Описание приложения

Используется `opencv` для чтения / записи `png`

Два класса - `MedianFilterSimple` и `MedianFilterOptimized`, соответствующие им функции

Выводится время, затраченное на исполнение функции

Пример запуска: `MedianFilter lena.png lena_filtered.png`




Вычислительная сложность

Размер картинки $N \times M$ и размер фильтра S (в данном случае равен 5)

Простой алгоритм:

Для каждого пикселя записываем данные окрестности в вектор длиной, сортируем, берем средний элемент, и записываем его в текущий пиксель.



	Простой алгоритм	Оптимизированный алгоритм
Асимптотика	$O(N * M * S^2 * \log(S^2))$	$O(N * M * S^2)$
Деления	$5 + 2 * S * S = 55$	0
Умножения	$S * S + 1 = 26$	0
Суммирования	$4 + 3 * S * S = 79$	$2 * S * S = 50$
Прибавление единицы	$S + S * S = 30$	$5 + 2 * S * S = 55$
Разница	$2 * S * S = 50$	0
Сравнения	$4 + S * S + S * S * \log S = 104$	$S * S + S * S = 50$
Время выполнения	1.3 сек	0.47 сек

Операции указаны для каждого пикселя, то есть нужно умножать на $N * M$

Из функций `std::sort` и `std::nth_element` учтены только количества сравнений



Оптимизированный алгоритм

Вектор был заменен на массив

Избегается аллокация памяти

Избегается вызов функции записи и чтения по индексу

Константы, которые могли быть вынесены, были вынесены. Таким образом, количество операций сократилось

`std::sort` была заменена на `std::nth_element`, что позволило сократить асимптотику относительно размера фильтра, и уменьшить количество операций