

Реализация алгоритма Stereo Block Matching на архитектуре SIMD и VLIW

Володин Вадим Евгеньевич

научный руководитель: ст.преп. А.А.Пименов

консультант: О. Окунев

Санкт-Петербургский Государственный Университет

21 Мая 2019, Санкт-Петербург

Stereo Block Matching

Алгоритм построения карты смещений

Подходит для работы в режиме реального времени

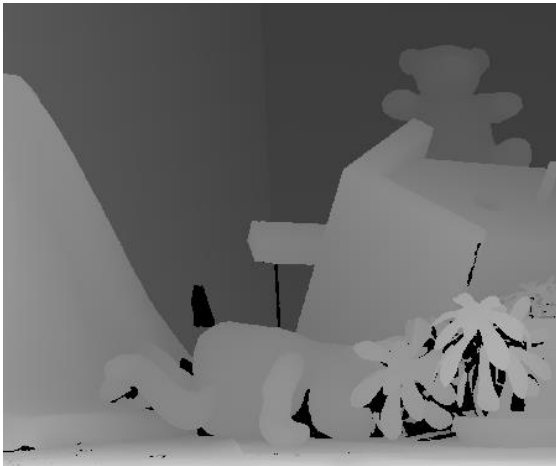
Stereo Block Matching



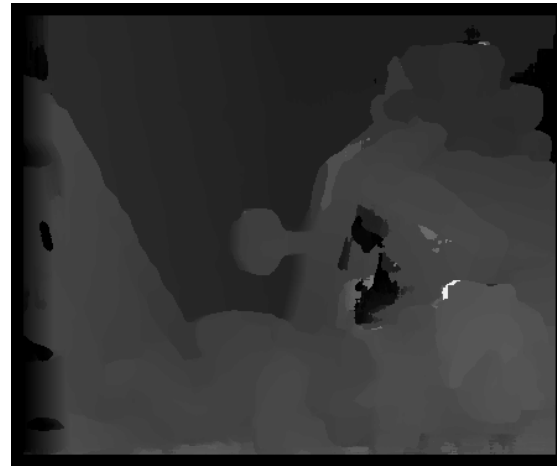
Left



Right



Truth



SBM

Stereo Block Matching

- Для каждого пикселя и каждого смещения подсчитывается метрика SAD (Sum of Absolute Differences – сумма модулей разности), выбирается смещение с минимальным значением

Архитектура процессора

Процессор EV6x, разработанный компанией Synopsys
SIMD, 512 бит

Доступные типы:

int16 – 16 чисел размера 32 бит

short32 – 32 числа размера 16 бит

char64 – 64 числа размера 8 бит

VLIW, 3 векторных слота, 1 скалярный

Объем векторной памяти 128 КБ

32 векторных регистра

Аккумуляторные регистры

Цель работы

Изучение алгоритма и архитектуры процессора

Реализация OpenCL kernel-а для данной архитектуры

Обзор аналогичных работ

Существуют реализации алгоритма для различных архитектур

Асимптотика

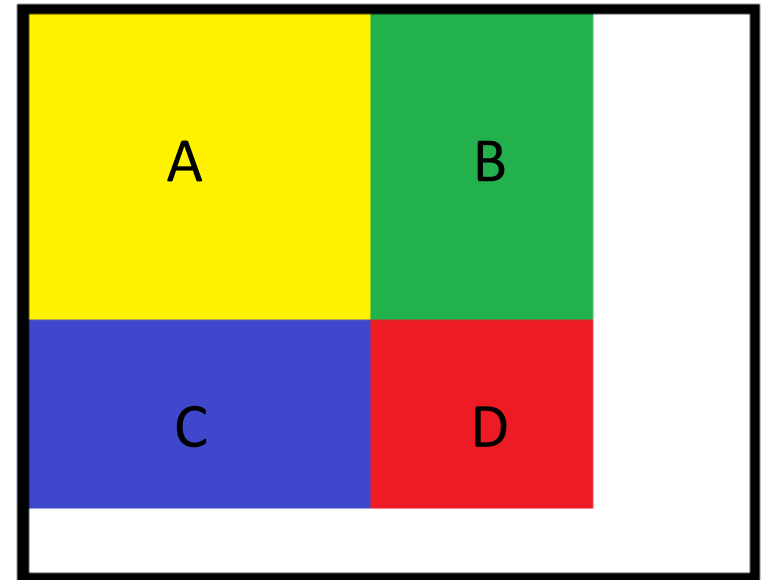
Алгоритм полного перебора

$$O(w * h * b^2 * d)$$

Алгоритм с улучшенной асимптотикой

$$O(w * h * d)$$

Существуют реализации для архитектуры SIMD



$$D = (A + B + C + D) - (A + C) - (A + B) + A$$

w, h - ширина и высота изображения, b - ширина окна, d - максимальное смещение

Технологии

- Язык С
- MetaWare OpenCL
- Тестовые данные vision.middlebury.edu

Результаты

Произведено избавление от reduce_min

Совершен переход от int16 к short32

Сокращено количество используемой памяти








Написан алгоритм полного перебора, алгоритм с улучшенной асимптотикой и алгоритм в конечном виде








Настроено тестовое окружение

Векторизация

- Значения SAD независимы для различных смещений

Избавление от reduce_min

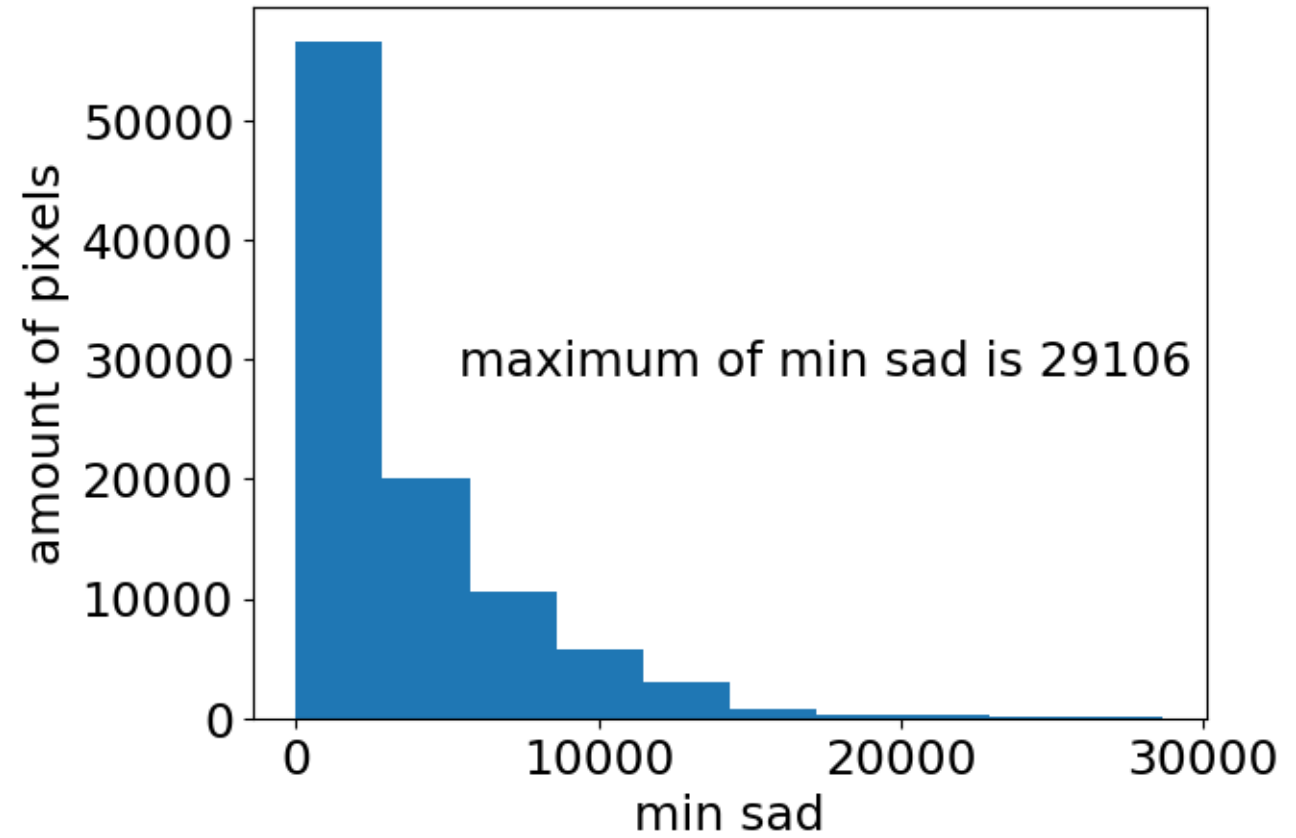
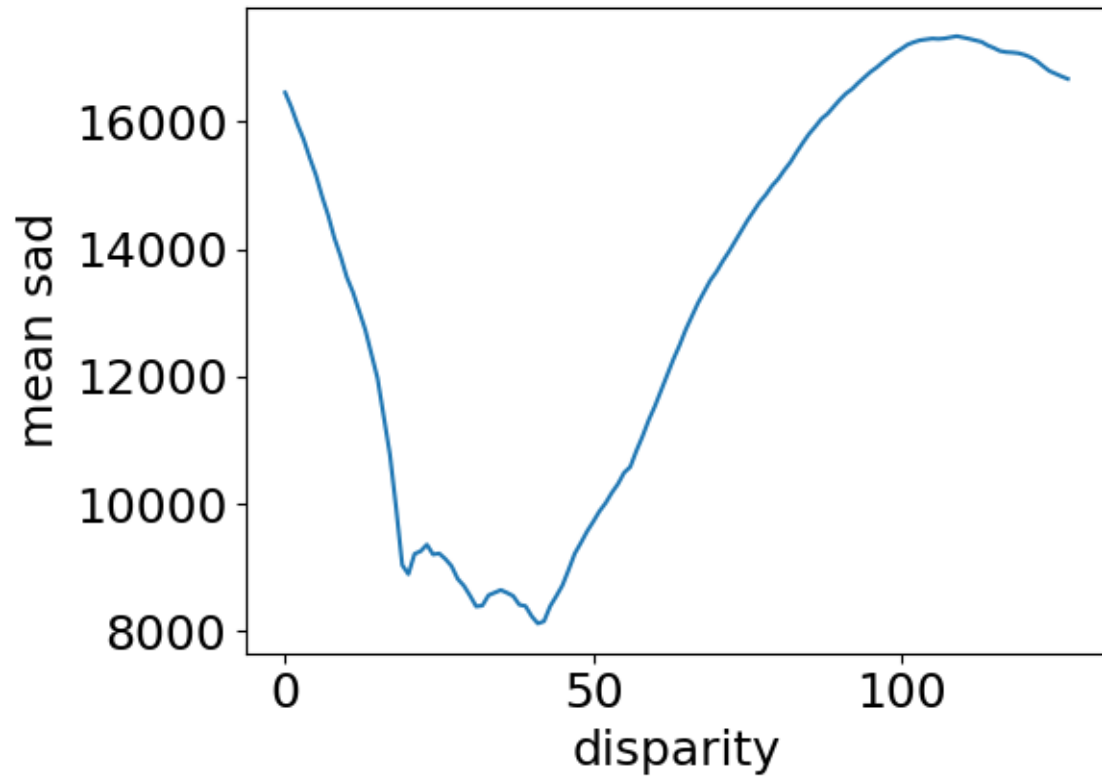
x \ d	0	1	2	3	4	5	6	
0								reduce_min
1								reduce_min
2								reduce_min
3								reduce_min
4								reduce_min
5								reduce_min
6								reduce_min

d \ x	0	1	2	3	4	5	6	
0								$sad[x] = now[x] < sad[x] ? now[x] : sad[x]$
1								$sad[x] = now[x] < sad[x] ? now[x] : sad[x]$
2								$sad[x] = now[x] < sad[x] ? now[x] : sad[x]$
3								$sad[x] = now[x] < sad[x] ? now[x] : sad[x]$
4								$sad[x] = now[x] < sad[x] ? now[x] : sad[x]$
5								$sad[x] = now[x] < sad[x] ? now[x] : sad[x]$
6								$sad[x] = now[x] < sad[x] ? now[x] : sad[x]$

int16 -> short32

- Размер окна по умолчанию – 21
- $\max \text{ SAD} = 21 * 21 * 255 = 441 * 255$
- $\max \text{ short} = 256 * 256 - 1$
- $\max \text{ SAD} < \max \text{ short} * 2$

int16 -> short32



Оценки

Количество используемой векторной памяти – 78 КБ

$$E = 4 * \max(\text{mem} + \text{shuffle}, (\text{mem} + \text{shuffle} + \text{alu})/3)$$

	shuffle	vmem	alu	mem+shuffle	total / 3	<i>Estimate, cycles per pixel</i>	<i>Performance</i>
reduce, int16	2	5	40	7	15.6	62.4	
reduce, short32	2	5	24	7	10.3	41.2	
no reduce, int16	4	9	14	13	9	52	
no reduce, short32	3	7	11	10	7	40	107