SIMD Optimization In Volume Rendering And Gaussian Filtering

(Rigorous Thesis)

Anton Vaško

Content

- Volume visualization
- Brief Overwiev of Volume Rendering Algorithms
- About SIMD
- SIMD Optimization in Ray Casting
- SIMD Optimization in Gaussian Filtering
- Conclusion

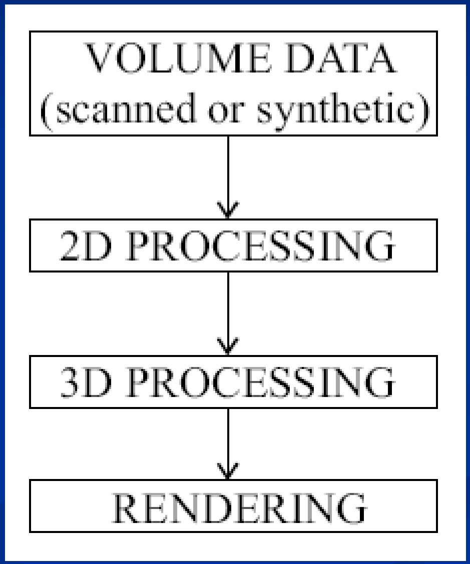
Quick Intro

- volume = 3D grid (discrete data)
- common in medical imaging (CT, MR, PET, SPECT ...)
- volumes can be large

  512 x 512 x 512 x 32bits = 512 MB

Volume visualization

- volume visualization:



- improving speed of rendering:

1. adjusting volume data
2. optimizing rendering algorithms

Adjusting volume data

- removing spurious particles
- reducing the number of non-air octree cubes
- achieved by smoothing – e.g. Gaussian Filtering

HW Volume Rendering

- volume = 3D texture
- volume rendering = mapping 3D texture on cube or GPGPU raycasting (shaders)
- advantages: pretty fast (all is done in HW)
- disadvantages: need for new (main-stream or better) graphics cards with big RAMs

SW Volume Rendering

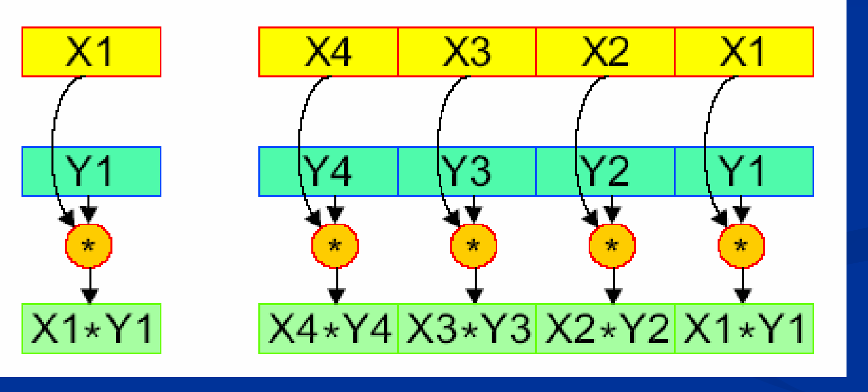
- more rendering techniques, e.g

1. Brute Force (ray casting)
2. Shear-Warp Factorization
3. Multiresolution Min-Max Octrees
4. and ... more ...

- disadvantages: not so fast, of course
- advantages: runs practically everywhere
- can be optimized for SIMD

SIMD (1)

- Single Instruction Multiple Data



- Extended CPU instruction set

SIMD (2)

- SIMD support on current CPUs:

1. Intel - MMX, SSE, SSE2, SSE3
2. AMD – MMX, 3DNow!, SSE
3. AMD64 – MMX, 3DNow!, SSE, SSE2, SSE3

- **Decision :**

  optimizing for 3DNow! and SSE

Approaches

- common aspect in volume rendering – processing a lot of data
- 2 basic approaches:

precalculate everything possible

vs

calculate everything on the fly

Present Time

- CPU is much faster than RAM
- calculating on the fly
- simple algorithms (ray casting)
- need for rewriting old algorithms

Decision:

SIMD Optimization of Ray Casting

SIMD in Ray Casting

Important parts:

- Bricking Data
- Entry Point Buffer
- Ray Casting

Bricking Data

- Non-linear storing in RAM
- Reasons:

1. CPU cannot read from RAM but from cache. Cache miss = <u>stall</u> for ≈40 clocks
2. Locality concept
3. Increased algorithm's data throughput with SIMD

Bricking Results

Table 5.2: *Impact of Bricking on Calculation Times [ms].*

B_x	B_y	B_z	EPB	RC-Init	RC-Accum	Total
256	256	128	9.94	15.27	1426.91	1452.12
128	128	128	7.33	11.08	743.73	762.14
64	64	64	7.28	10.13	148.25	165.66
32	32	32	8.93	9.67	73.95	92.55
16	16	16	18.72	9.62	54.07	82.41
8	16	16	30.34	9.88	51.89	92.11

Entry Point Buffer

- Similar to Depth buffer
- Parallel projection → rasterizing just one block and copying
- Doing *memset* and *memcpy* is 'delicacy' for SIMD

EPB Results

Table 5.4: *Optimization of stage EPB.*

PC	Duration [ms]			Speedup	
	None	3DNow	SSE	3DNow	SSE
PC1	19.13	14.4	-	1.33	-
PC2	48.9	40.2	-	-	1.22
PC3	26.39	23.0	22.7	1.15	1.16

Ray Casting

- 2 parts:

1. Initializing rays according Entry Point Buffer → SIMD optimizable
2. Processing rays (accumulation):
   not 'SIMD optimization friendly'

Ray Casting Results

Table 5.6: *Optimization of the Ray Initialization stage.*

PC	Duration [ms]			Speedup	
	None	3DNow!	SSE	3Now!	SSE
PC1	9.5	6.0	-	1.6	-
PC2	22.0	-	16.0	-	1.4
PC3	10.03	7.06	7.05	1.4	1.4

Ray Casting Results

Table 5.7: *SIMD optimization of the Accumulation stage.*

PC	Duration [ms]			Speedup	
	None	3DNow!	SSE	3Now!	SSE
PC1	55.7	54.7	-	1.02	-
PC2	122.1	-	116.3	-	1.05
PC3	53.53	52.23	50.7	1.02	1.06

SIMD and Gaussian Filtering

- separable – 3 passes
- symmetric – saving multiplications
- floating-point
- filtering in-place
- without temporary buffer – how?

Extended volume 1

- new technique - extended volume

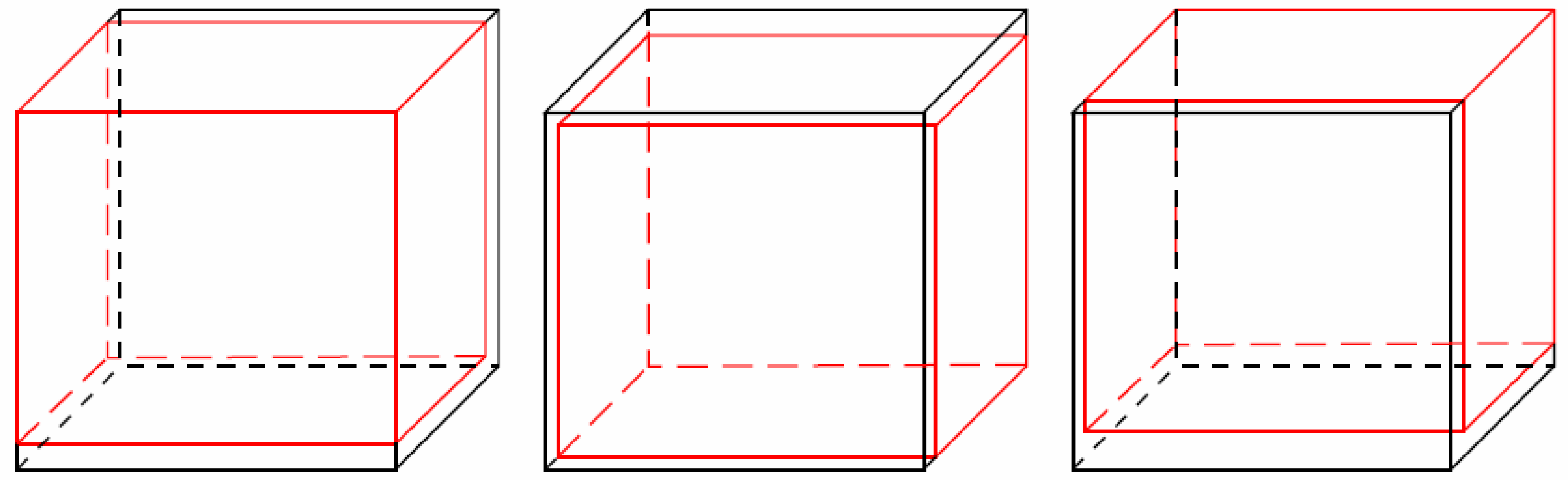


Figure 4.1: *Position of the original volume in the extended volume during filtering. Left - at the beginning before filtering. Middle - after FilterX. Right - after FilterY. After FilterZ the filtered volume is positioned again in the top left of the extended volume .*

Extended volume 2

- reducing passes

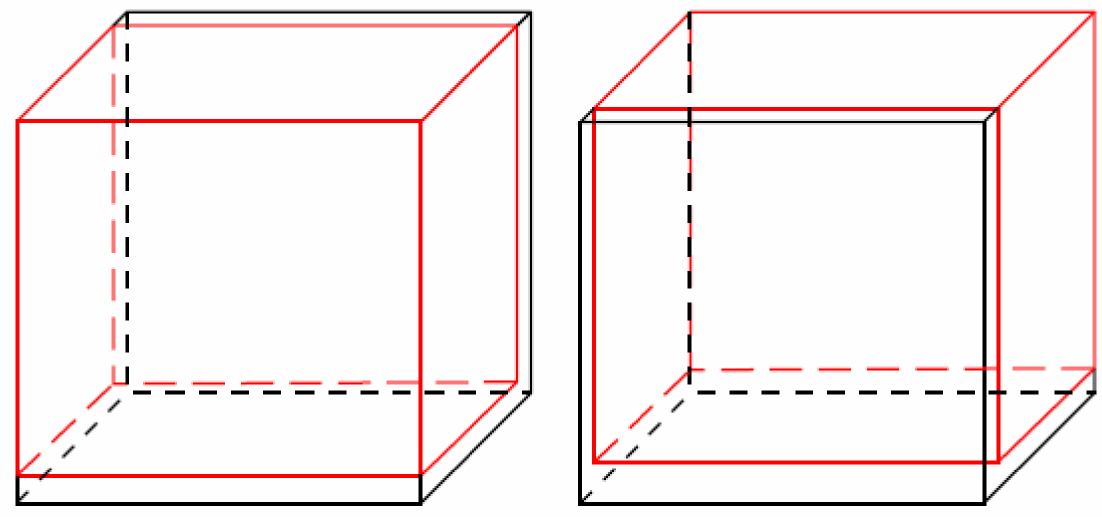


Figure 4.2: *Position of the original volume in the extended volume during filtering. Left - at the beginning before filtering. Right - after FilterXY. After FilterZ the filtered volume is positioned again in the top left of the extended volume .*

- parallelization of computation for SIMD

Limitations

- number of SIMD registers – 8 (x86)
- extended volume – additional extended slices
- Decision – 3 additional slices, 3 additional lines for each slice =>
- Gauss3, Gauss5, Gauss7

Results

Table 6.9: *Results of gauss3 on PC3.*

Total	Duration [s]			Speedup	
	None	3DNow!	SSE	3Now!	SSE
volume1	0.1035	0.0051	0.0047	20.24	21.88
volume2	0.8517	0.0442	0.0498	19.26	17.11
volume3	6.9225	0.4362	0.5036	15.87	13.75
volume4	379.7929	47.1538	32.2716	8.05	11.77

Conclusion

- Real time volume rendering – GPU
- Quality volume rendering – CPU (12% speedup with SIMD)
- Gaussian Filtering – positively SIMD
- SIMD Optimization principles can be often successfully used even when not immediately followed by assembler implementation! (bricking, ...)

Future work

- Optimization of filtering:

1. Gauss9, Gauss11, Gauss13, Gauss15 (16 SIMD registers in x86-64)
2. General separable filtering (not only Gaussian, no kernel size restriction)
3. Nonseparable filtering

Thanks for your attention !

vasko.anton@gmail.com