Evaluation of Rendering Optimizations for Virtual Reality Applications in Vulkan

Paul Preißner

5 March 2020

Final: Master Informatics: Games Engineering

Supervisor: Prof. Gudrun Klinker, Ph.D.

Advisor: M.Sc. Sven Liedtke

in collaboration with RTG Echtzeitgraphik GmbH

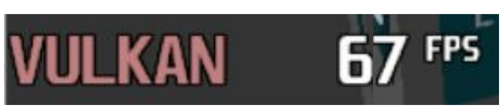
Introduction / Motivation

- Collaboration with RTG Echtzeitgraphik GmbH (@gate)
- Hot topics: Vulkan and Virtual Reality
- High performance rendering required for many applications/games
- Available material on optimization in Vulkan rather scarce, material on optimization for VR often basic
- **Especially in enterprise**: custom solutions wanted, licensed engines may be a problem

Goals of this Thesis

- in-house engine (RTG Tachyon) as foundation
- implementation of several optimizations for VR rendering
  - pre-render input reduction
  - render effort reduction
- gain insight into performance impact through benchmarking and evaluation
- recommendations based on these results

- ideal showcase: complex, high object & poly count scene goes from stuttering to smooth VR



→

Research Issues

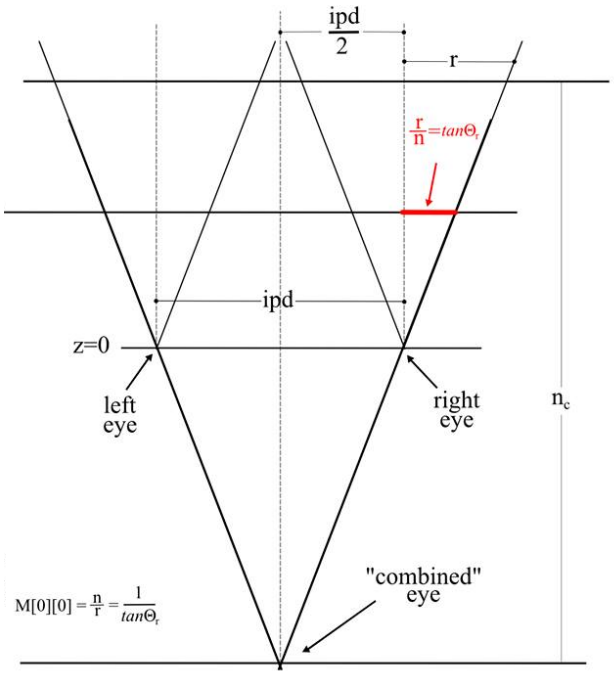
- Basic premise: “how to make *VR* rendering fast?”
- many generic approaches exist, but few dive into Vulkan specifics
  → which potential approaches are *designed* for VR?
  → which can be combined for greater impact?
  → how do they impact performance for a large, complex scene?
  → what to watch out for when building these on Vulkan?

Related Work (selected)

- Vulkan community contributions
  - S. Willems. *Vulkan C++ examples and demos*. 2015-2019
  - A. Kapoulkine. *Niagara*. 2018
- C. Everitt's general *VR contributions*
- N. Whiting. *Oculus Connect 4 | Technical Postmortem for Robo Recall: Superfrustum culling*. 2017
- A. Vlachos. *GDC2015: Advanced VR Rendering*. 2015
- R. Palandri, S. Green. *Hybrid Mono Rendering*. 2016
- D. Di Donato, R. Palandri, and R. Vance. *High quality mobile VR with Unreal Engine and Oculus*. 2017
- Nvidia *VRWorks* & AMD *LiquidVR*

etc.

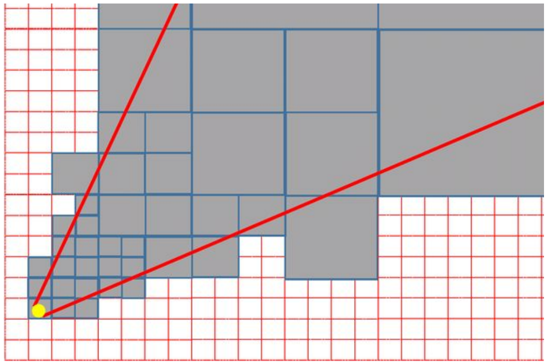
Proposed Work [planned]



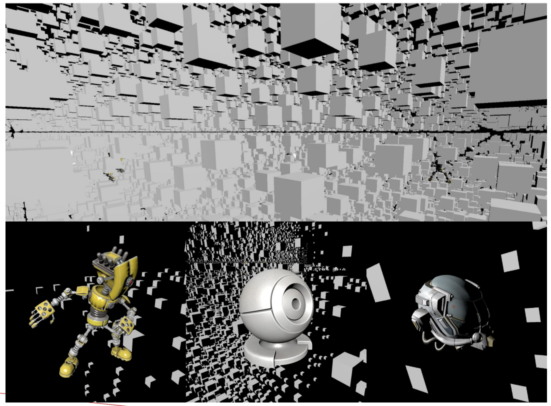
Cass Everitt 2015

- in Tachyon renderer, integrate
  - Multiview Stereo rendering
  - Hierarchical (Super)Frustum Culling
  - (Masked) Software Occlusion Culling
  - Monoscopic Far-Field rendering
  - Round Robin VR occlusion
- benchmark individual & combined performance & resource impact using two test environments
  - simple high primitive count scene
  - complex "real-world" showcase scene (provided from real enterprise project)
- outline additional viable approaches for reference/completeness

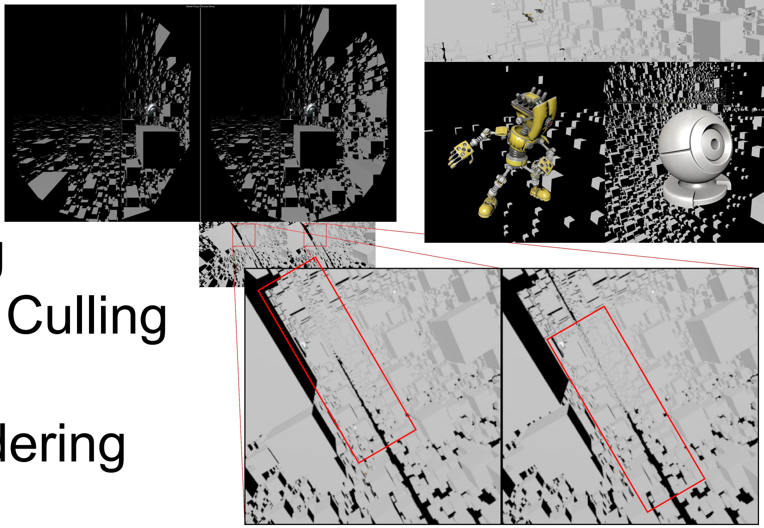
Oscar Martinez Rubi 2015



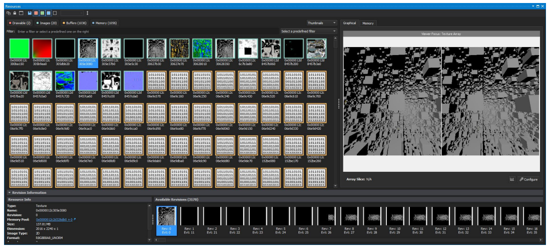
Proposed Work [actual]



- in Tachyon renderer, integrate
  - Multiview Stereo rendering
  - Hierarchical Superfrustum Culling
  - Fitted Stencil Masking
  - Monoscopic Far-Field rendering



- benchmark individual & combined performance impact & resource usage
  - *one* test environment: synthetic high object/tri count scene
- outline additional viable approaches for reference/completeness



Discussion of Potential Issues

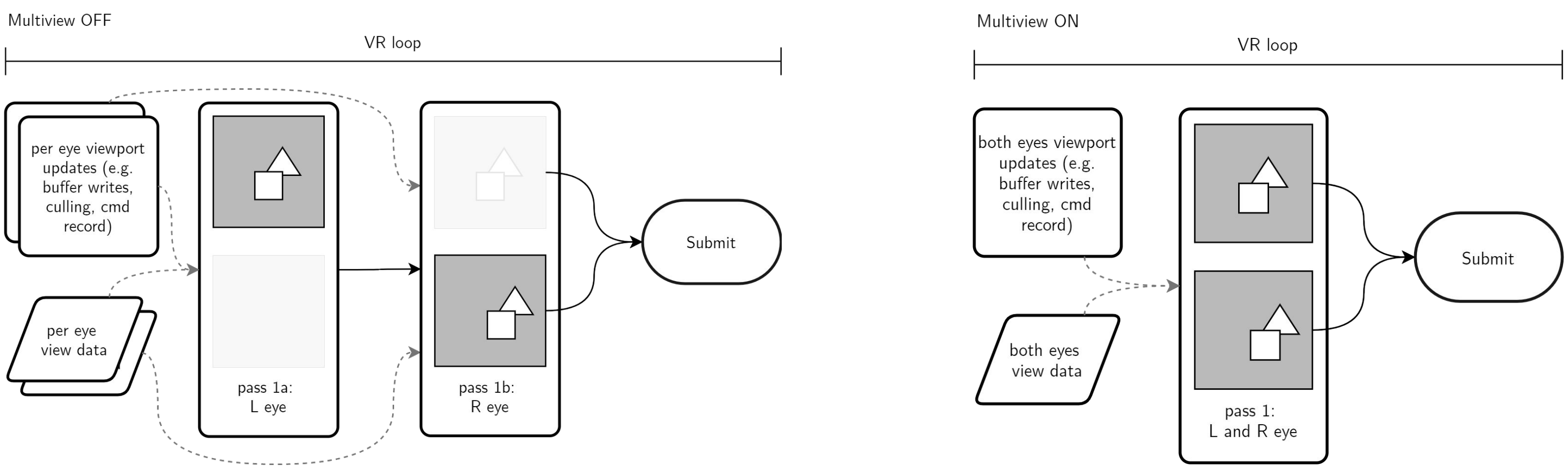
- Risk: implementation difficulty
  - started at virtually zero low-level graphics/Vulkan programming experience
- Risk: unexpected performance stagnation/regression due to unforeseen limitations (bad test scene, flawed implementation, API quirks, etc)
- Risk: enterprise involvement, some code may be under NDA

Discussion of Actual Issues

- Risk: implementation difficulty
  → first month lost working on from-scratch renderer
  → one month spent on from-scratch culling system
- Risk: performance stagnation/regression due to unforeseen limitations
  → badly ordered render passes, leading to low shader utilization (see benchmark results)
- Risk: enterprise involvement, some code may be under NDA
  → not an issue, all code in-house

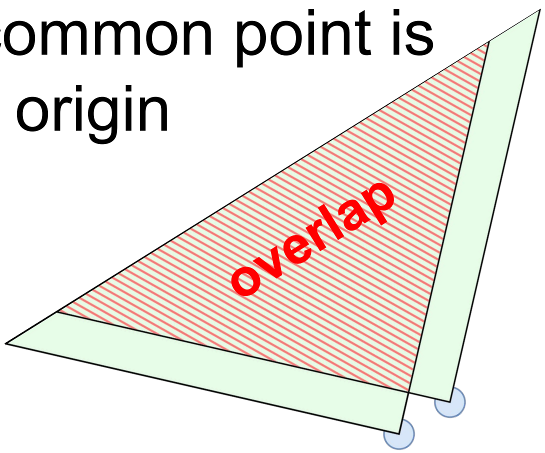
Impl.: Multiview Stereo Rendering

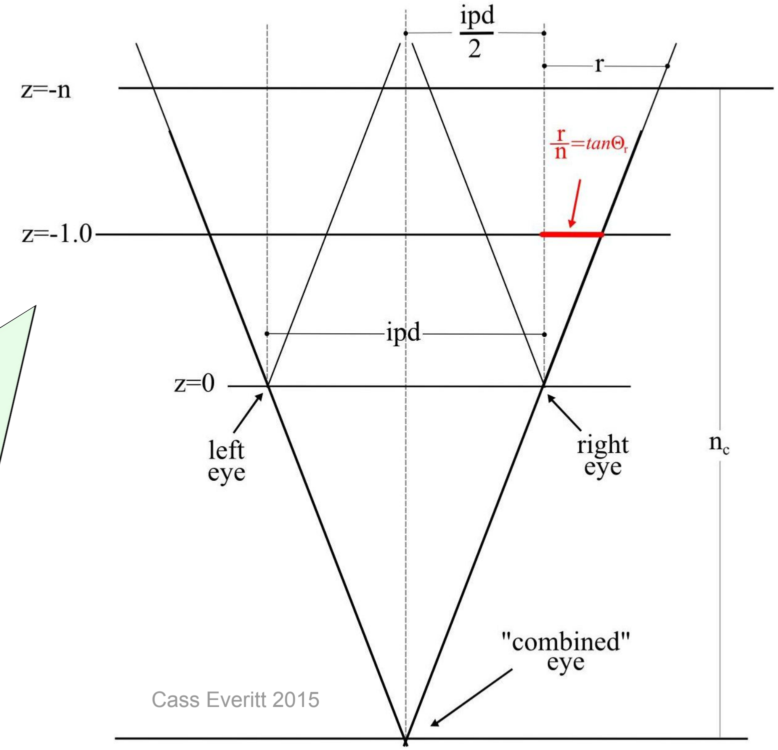
- submit draw commands for both eyes in one call instead of two → halves submission cpu-time
- on supported GPU architectures, e.g. NV Turing: hardware acceleration → skipping eye-independent pipeline stages (vertex/tess/geo), reduces gpu-time
- Vulkan extension VK_KHR_Multiview, modified/merged VkRenderPass, single VkCommandBuffer recording



Impl.: Superfrustum Culling

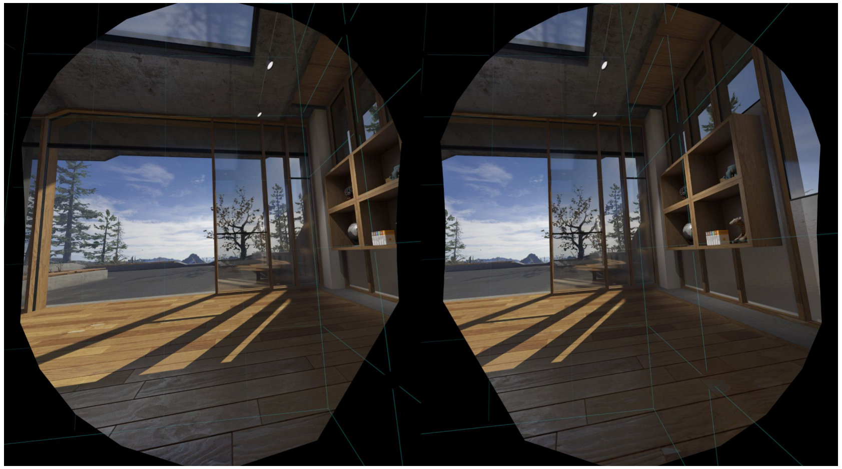
- large overlap between wide angle eye frusta
  → geometrically construct single frustum around stereo frusta → reduced number of culling/intersection checks
- symmetric normalization of asymmetric eye frusta
- widest half opening angles a.k.a. opening tangents
  → tangents' common point is Superfrustum origin





Impl.: Fitted Stencil Masking

- circular overlay mask on each eye
  - in *stencil* buffer: pixel-perfect mask to fail the stencil test, or
  - in *depth* buffer: first draw mesh mask to fail depth/early Z test
- subsequent draws skip mask area in fragment stage
- HMD-fitted mask can be queried from OpenVR via GetHiddenAreaMesh(...), but not all HMDs offer one!
- here: stencil buffer mask queried and drawn at startup, reused every frame
- Vulkan stencil operation states allow ops and access masks tailored for each VkPipeline



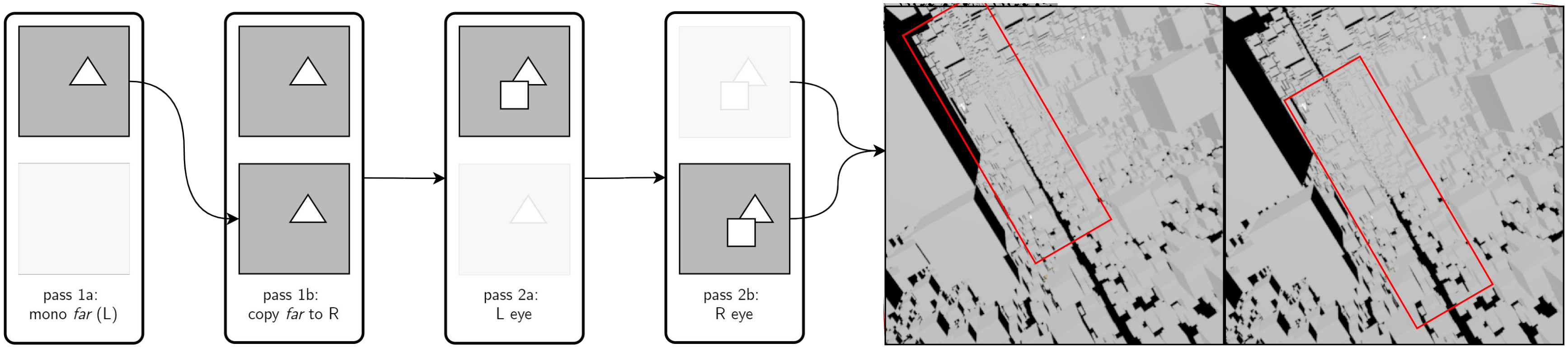
Impl.: Monoscopic Far-Field Rendering

- insignificant (or even <1px) stereo separation for distant objects → render near-field as normal stereo, far-field only mono, then composite → may cut down on draw calls
- inconsistent impact, difficulty with post-processing effects → abandoned by Oculus & Epic

- Tachyon uses additional set of VkRenderPass, draw-call VkCommandBuffers, pass synchronization VkSemaphores, GPU-side camera data, dedicated image transition and L-to-R-copy VkCommandBuffer, additional culling frustum & pass

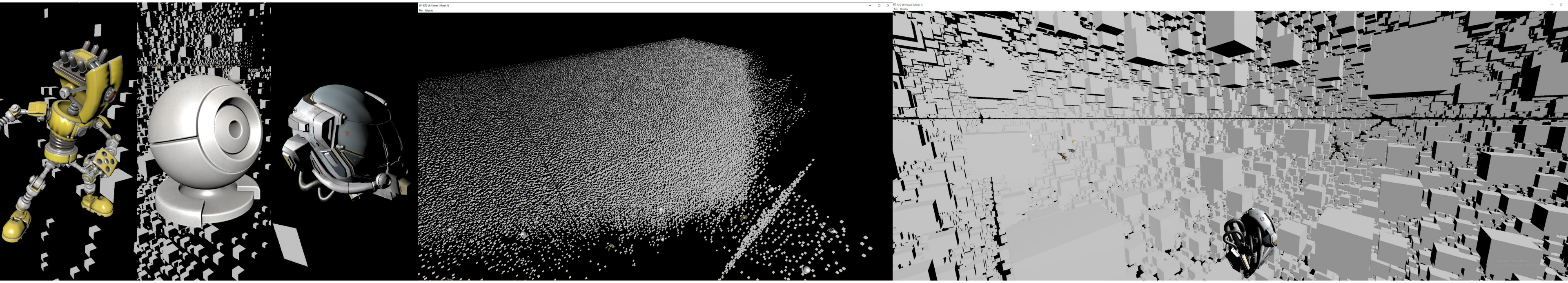
Impl.: Monoscopic Far-Field Rendering

- Tachyon MFFR renders far-field first, stereo on top
  → no memory overhead, but worse pipeline saturation
- Far-Field projection matrix constructed incorrectly
  → bad shift on both screen-space axes
- Incorrect culling result sets → severe overdraw and performance loss



Evaluation (Benchmark setup)

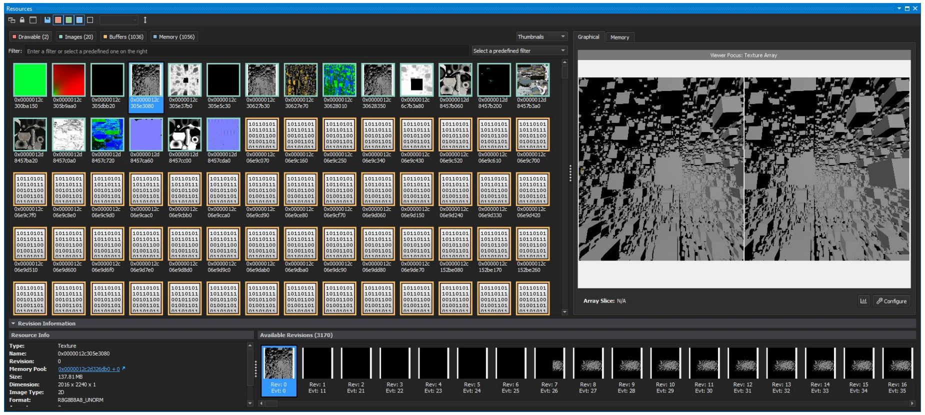
- synthetic test scene: dense “cloud” of primitives plus sparse grid of high-poly objects (~11.1m total)
- fixed camera track for repeatability
- 10 loops per configuration (54000 frames), averaged to single result loop
- test machine: i7 6700, 32GB, RTX 2080, Valve Index



Video memory usage

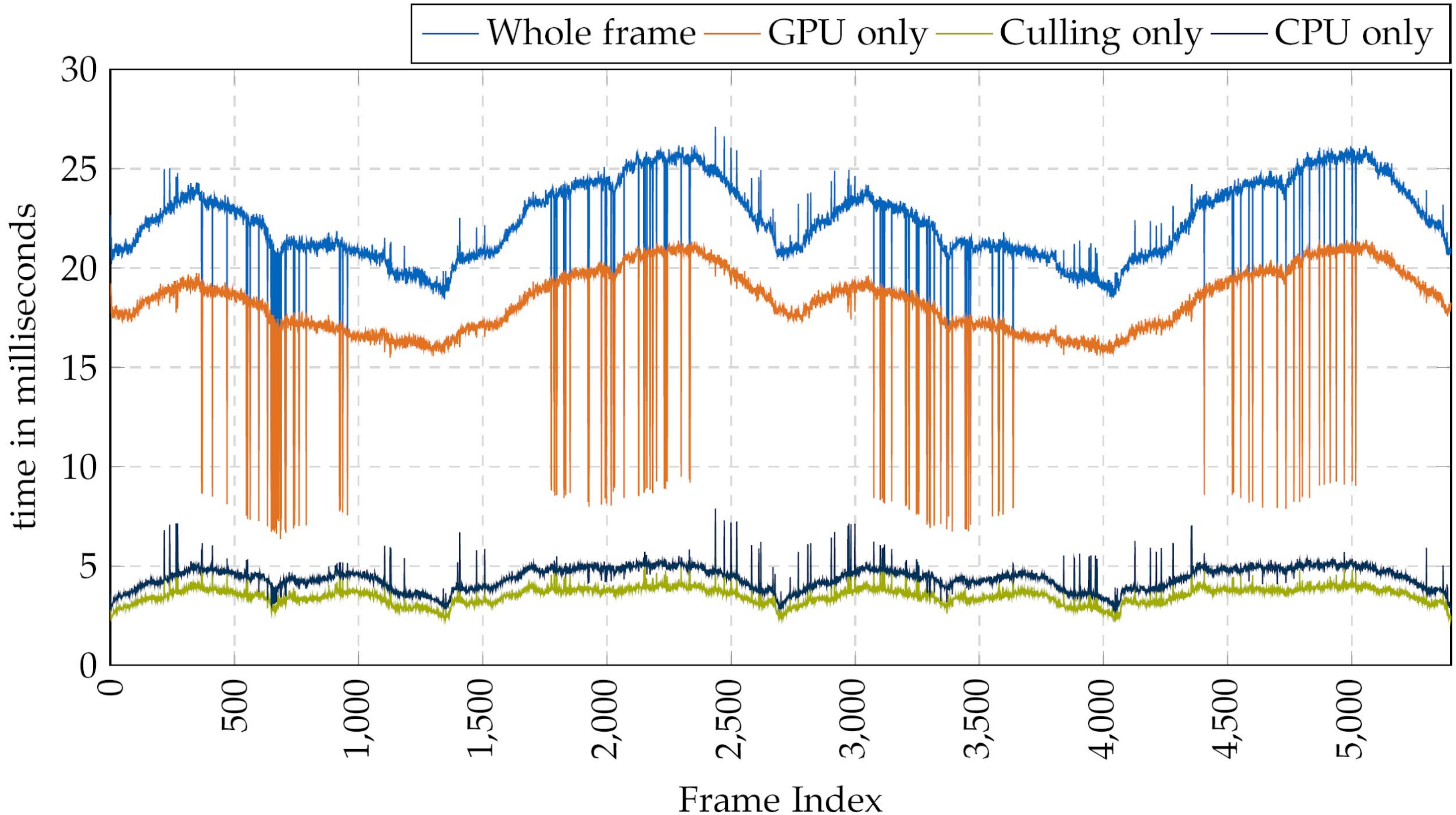
- for all cases except Stencil Masking:
  - 678MB object model matrices
  - 277MB (color + depth) framebuffer
  - <60MB textures, index buffers
  - 257MB staging (only at startup)
- with Stencil Masking: +140MB framebuffer

→ no major vmem impact from optimizations unless assets dynamically loaded/unloaded (not in Tachyon)



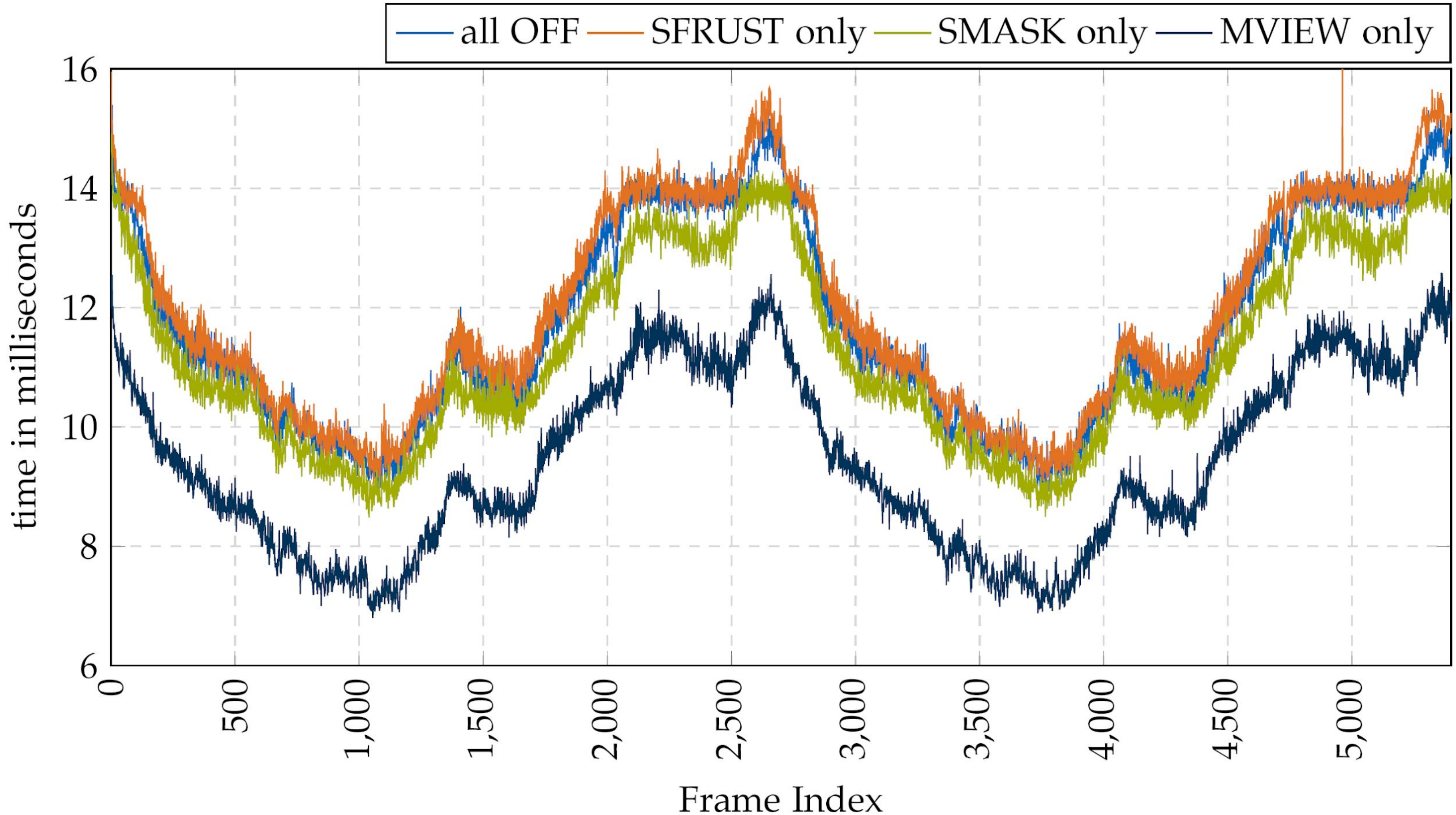
Benchmark results

- MFFR performance issue



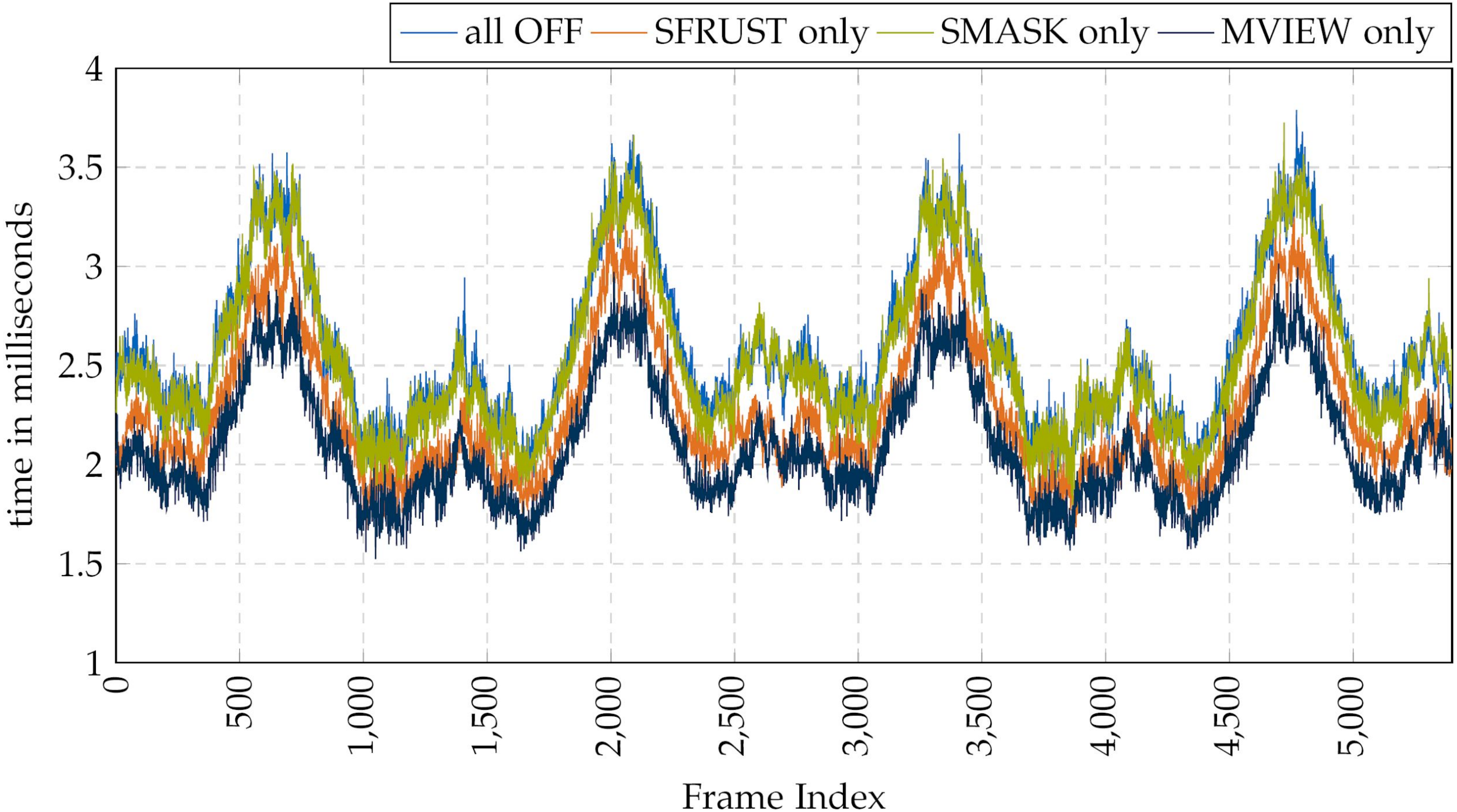
Benchmark results

- Individual optimizations (Whole frame)



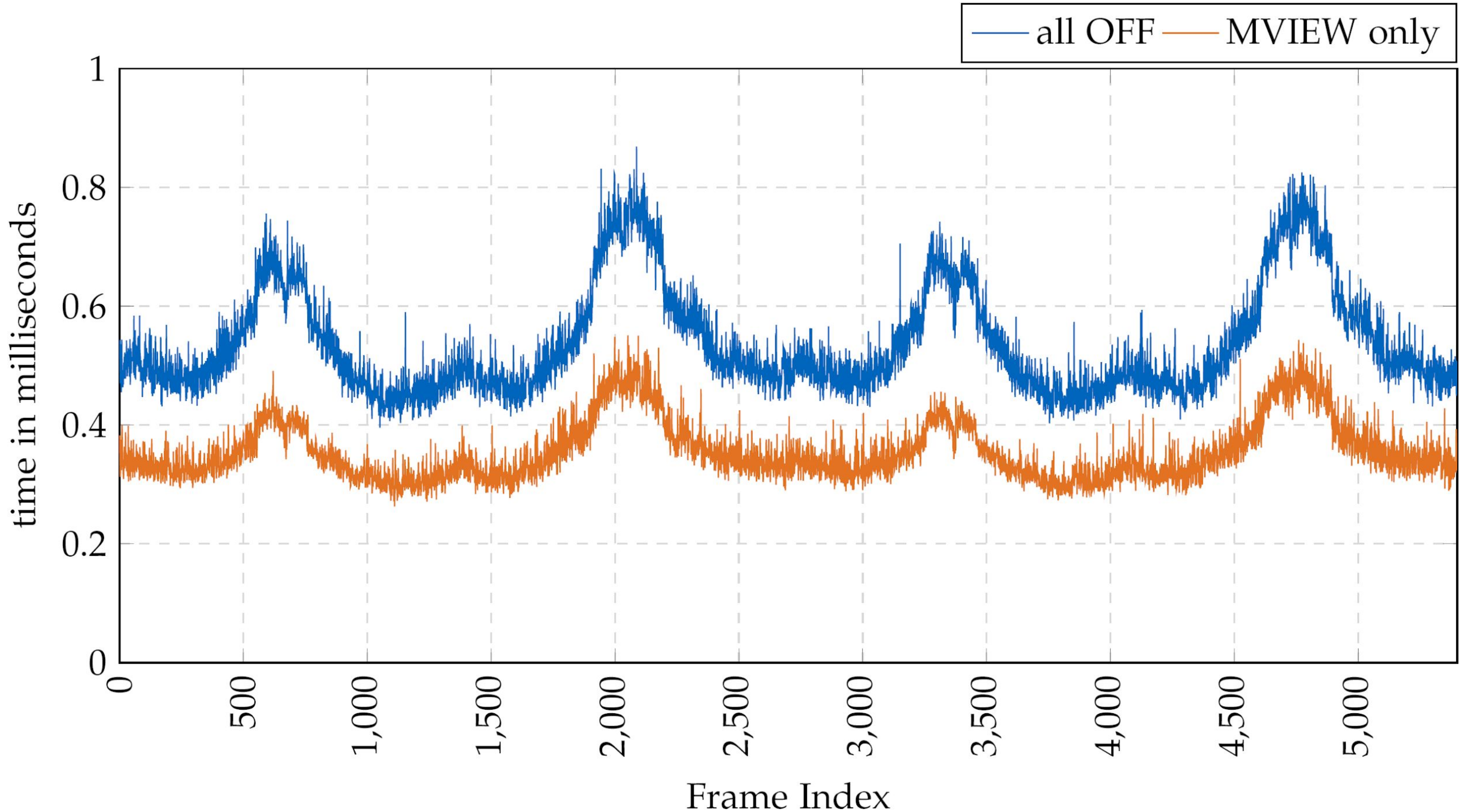
Benchmark results

- Individual optimizations (CPU time)



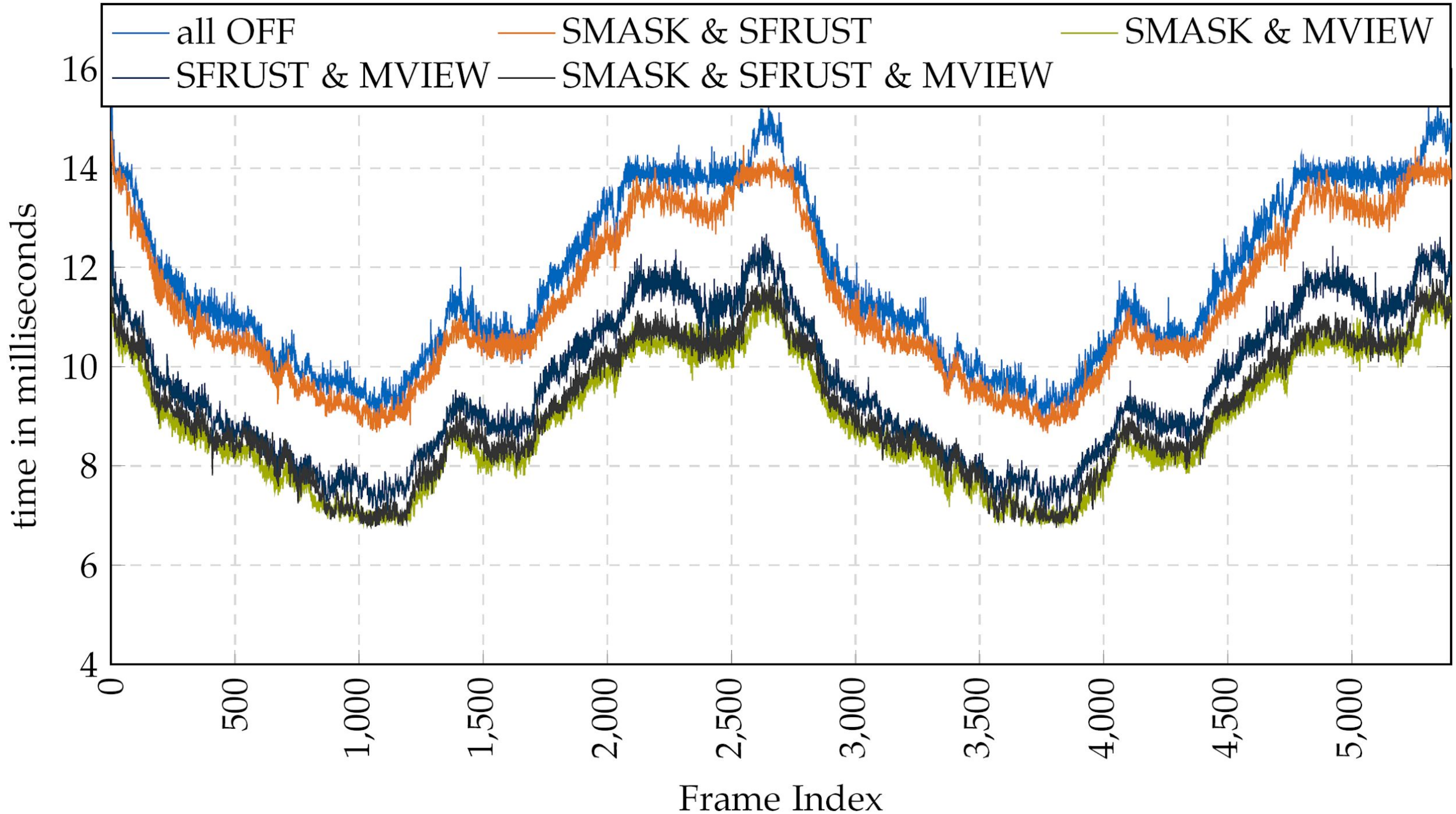
Benchmark results

- Multiview cpu-time (sans culling) vs baseline

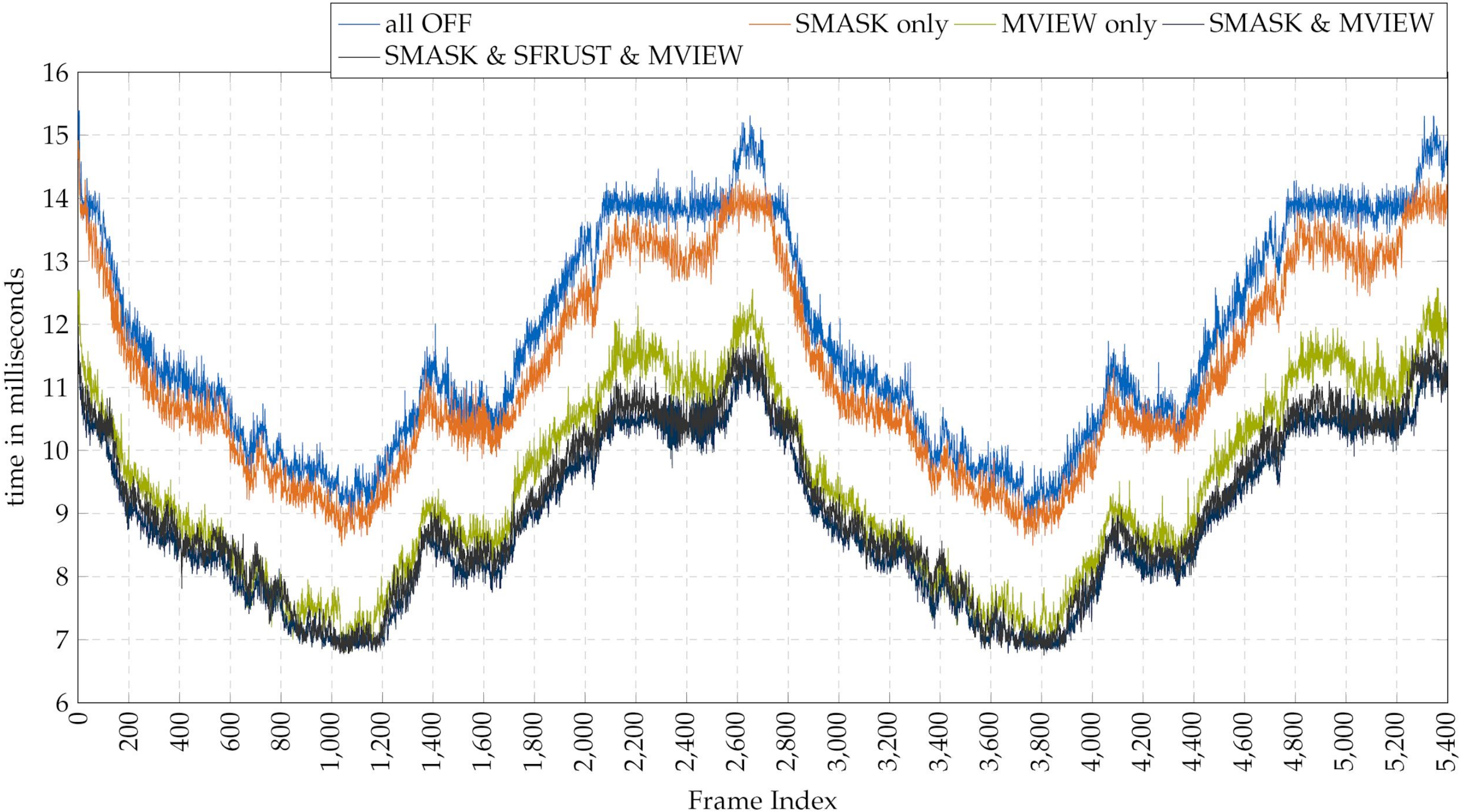


Benchmark results

- Combined optimizations (Whole frame)



Most viable configurations



Discussion / Suggested Future Work

Gist:

- Multiview a must-have, especially when hw-accelerated
- Stencil Masking a simple, cheap must-have
- Superfrustum only viable in conjunction with Multiview
- Superfrustum and sw-Multiview necessary when CPU power is scarce

Future:

- MFFR broken, when fixed likely only valuable on mobile → may not be worth the effort
- Tachyon to be refined & extended with other referenced approaches, e.g. Fixed Foveated Rendering
- Lengthier (approach-specific) Vulkan implementation details to be provided publicly

Conclusion

- Valuable insight, even if some parts unsuccessful
- Significant performance gains to be had for VR apps, crucial for good performance in graphically intense scenes
- Vulkan programming requires much caution, but rewards are worth it

Q&A

Stereo Multiview references

- S. Bhonde and M. Shanmugam. T*uring Multi-View Rendering in VRWorks | NVIDIA Developer Blog*. 2018. url: https://devblogs.nvidia.com/turing-multi-view-rendering-vrworks/
- K. Jez. GPUOpen: *AMD LiquidVR MultiView Rendering in Serious Sam VR*. 2017. url: https://gpuopen.com/amd-liquidvr-multiview-rendering-in-serious-sam-vr/
- F. Serrano. *Multiview on WebXR*. 2019. url: https://blog.mozvr.com/multiview-on-webxr/

Superfrustum references

- C. Everitt. *single combined camera matrix*. 2015. url: https://scontent-dus1-1.xx.fbcdn.net/v/t31.0-8/11334168_10154006919426632_2185539868454578065_o.jpg?_nc_cat=107&_nc_ohc=QEbGuUs7xiIAX_q3hjC&_nc_ht=scontent-dus1-1.xx&oh=4c8cb329ef851d5ab51161943910fed9&oe=5ECFE2F1
- C. Everitt. *asymmetric eye matrix normalization*. 2015. url: https://scontent-dus1-1.xx.fbcdn.net/v/t31.0-8/10460839_10154007978676632_3794989256420316318_o.jpg?_nc_cat=110&_nc_ohc=JuTLkrDwfT0AX_uUssk&_nc_ht=scontent-dus1-1.xx&oh=a3837266309d26af90584470482a1fea&oe=5EC2AD91
- N. Whiting. *Oculus Connect 4 | The Road to Shipping: Technical Postmortem for Robo Recall: Superfrustum culling*. 2017.
- V. Oddou. *VR and frustum culling - Computer Graphics Stack Exchange*. 2017. url: https://computergraphics.stackexchange.com/a/4765

Stencil Masking references

- A. Vlachos. *GDC2015: Advanced VR Rendering*. pp. 51–65. 2015.
- J. de Vries. Learn OpenGL: Stencil testing. 2014. url: https://learnopengl.com/index.php?p=Advanced-OpenGL/Stencil-testing

MFFR references

- R. Palandri and S. Green. *Oculus Developer Blog: Hybrid Mono Rendering in UE4 and Unity*. 2016. url: https://developer.oculus.com/blog/hybrid-mono-rendering-in-ue4-and-unity/
- *Monoscopic Far Field Rendering*. 2016. url: https://docs.unrealengine.com/en-US/Platforms/VR/DevelopVR/MonoFarFieldRendering/index.html

More references

- *VRWorks - Multi-Res Shading*. 2016. url: https://developer.nvidia.com/vrworks/graphics/multiresshading
- *VRWorks - Variable Rate Shading (VRS)*. 2018. url: https://developer.nvidia.com/vrworks/graphics/variablerateshading
- R. Palandri, S. Gosselin, and C. Pruett. *Oculus Developer Blog: Optimizing Oculus Go for Performance*. 2018. url: https://developer.oculus.com/blog/optimizing-oculus-go-for-performance/
- *Foveated Rendering on the VIVE PRO Eye*. 2019. url: https://zerolight.com/news/tech/foveated-rendering-on-the-vive-pro-eye
- U. Haar and S. Aaltonen. *SIGGRAPH 2015: Advances in Real-Time Rendering in Games: GPU-Driven Rendering Pipelines*. pp. 10–25. 2015.
- A. Vlachos. *GDC2016: Advanced VR Rendering Performance*. 2016.