High-Performance Graphics 2021
N. Binder and T. Ritschel
(Guest Editors)

Volume 40 (2021), Number 8

A Halfedge Refinement Rule for Parallel Catmull-Clark Subdivision Supplemental Material: GPU Performance Measurements

J. Dupuy and K. Vanhoey

Unity Technologies

This document provides exhaustive performance measurements of open source GPU-based Catmull-Clark subdivision implementations.

Methods. The open source implementations we identified and consider are listed in Table together with the type of mesh features they do or do not handle. **OpenSubdiv** is the industry standard. We run it using its GLSL backend on Linux. **Nießner *et al.***'s application is Windows-only (it uses DirectX). **Patney *et al.*** provides an end-to-end rendering implementation in CUDA, that does not allow to isolate **Vertex Point Subdivision**. Thus we only execute it for the **End-to-End Subdivision** scenario. For fair comparison, we modified their code to output a uniform instead of an adaptive subdivision. **Mlakar *et al.*** provide an end-to-end parallel GPU implementation using CUDA. To measure timings for the *Vertex Point Subdivision* scenario, we modified their code to pre-compute topology subdivision and store its results in a table. We then run and time *Vertex Point Subdivision* only. We report runtimes as measured by their provided CUDA timers, which we modified in one respect: Their count excludes time spent on CUDA memory allocations and free's (because they can in theory be pre-allocated) but also memset operations (which cannot be preallocated) which we included. **Our** method provides both scenario's using GLSL shaders as illustrated in the accompanying code.

Method	O.S.	Non-quad	Creases
OpenSubdiv [Pix13]	Linux	yes	yes
Nießner *et al.* [NLMD12]	Windows	yes	no
Patney *et al.* [PEO09]	Windows	no	no
Mlakar *et al.* [MWS*20]	Linux	yes	no
Ours	Linux	yes	yes

Data. We consider a total of 8 meshes with different properties (see sections 1 to 8). The first four meshes have semi-sharp creases, which are only supported by OpenSubdiv and our implementation. The last four consist of two quad-only and boundary-free meshes and two more complex meshes with boundaries and non quads.

Protocol. We performed GPU runtime measurements on each mesh and for six subdivision depths (depths 1 to 6) by all methods capable of handling them. Note that subdividing the T-Rex model (Section 8) down to depth 6 using OpenSubdiv resulted in a GPU out-of-memory, hence its absence in the plots. For each subdivision, we show three plots, akin to those in our main paper: two show the timings for the *End-to-End Subdivision* and *Vertex Point Subdivision* scenarios, respectively, and one shows the timings of each of our individual GLSL shaders. Each plot reports the median runtime measured over 50 evaluations and the minimum and maximum runtime as error bars. As explained in the main paper: we made sure all timings include shader/kernel execution time, necessary memset instructions, state changes, and CPU-GPU synchronizations. All measurements were done on an NVIDIA RTX 2080 graphics card and a 4.00GHz Intel Core i7-8086K CPU with 32GiB RAM.

Discussion. Results are all in support of our analysis of Section 6 in the main paper. We additionally note that we observe large variations among runs (see error bars) for the method of Nießner *et al.*

References

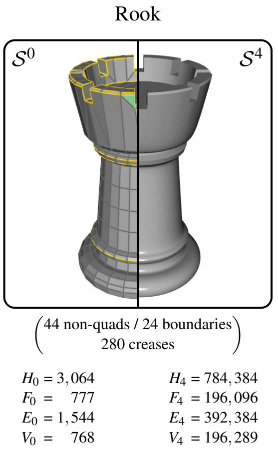
[MWS*20] Mlakar D., Winter M., Stadlbauer P., Seidel H.-P., Steinberger M., Zayer R.: Subdivision-specialized linear algebra kernels for static and dynamic mesh connectivity on the gpu. *Computer Graphics Forum 39*, 2 (2020), 335–349. 1, 6, 7, 8, 9

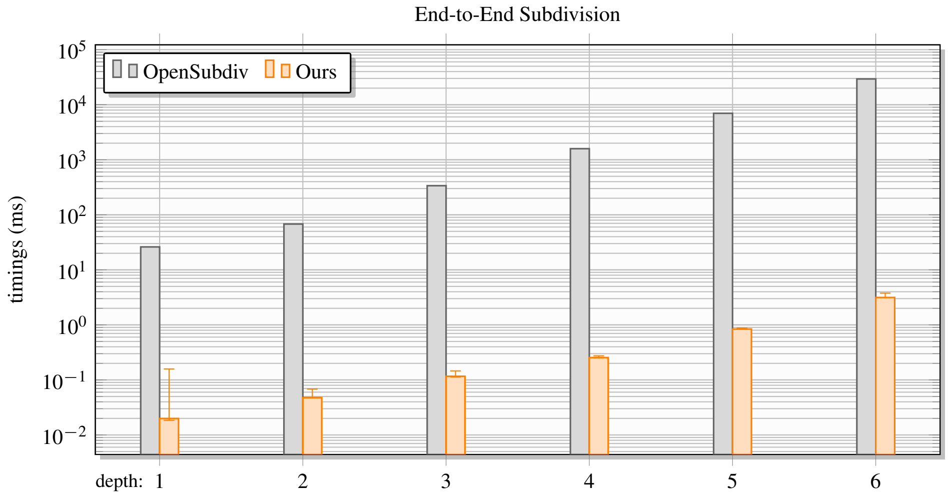
[NLMD12] Niessner M., Loop C., Meyer M., Derose T.: Feature-adaptive gpu rendering of catmull-clark subdivision surfaces. *ACM Trans. Graph. 31*, 1 (Feb. 2012). 1, 6, 7, 8, 9

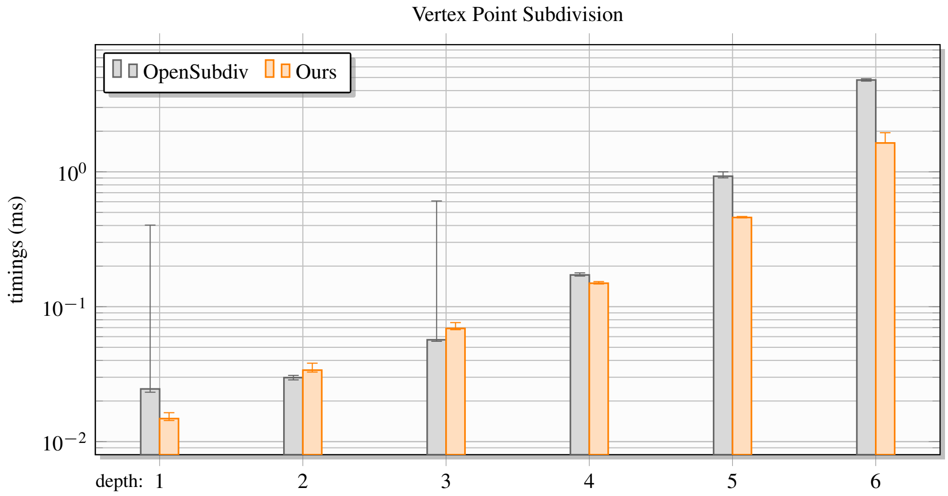
[PEO09] Patney A., Ebeida M. S., Owens J. D.: Parallel view-dependent tessellation of catmull-clark subdivision surfaces. In *Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), HPG '09, Association for Computing Machinery, pp. 99–108. 1, 6, 7

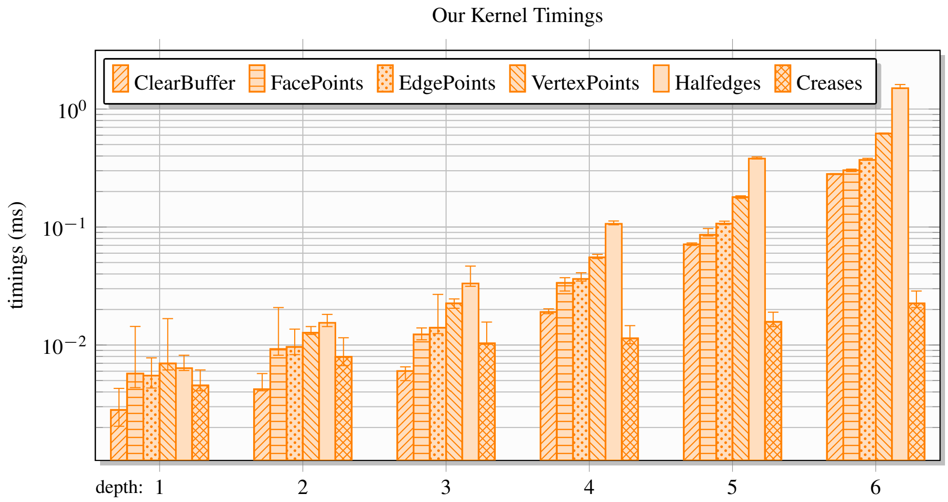
[Pix13] Pixar: Opensubdiv from research to industry adoption. In *ACM SIGGRAPH 2013 Courses* (New York, NY, USA, 2013), SIGGRAPH '13, Association for Computing Machinery. 1

1. Rook


Rook
$\mathcal{S}^0$
$\mathcal{S}^4$
44 non-quads / 24 boundaries
280 creases
$H_0 = 3,064$
$F_0 = 777$
$E_0 = 1,544$
$V_0 = 768$
$H_4 = 784,384$
$F_4 = 196,096$
$E_4 = 392,384$
$V_4 = 196,289$

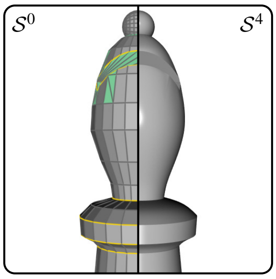

End-to-End Subdivision
OpenSubdiv
Ours
timings (ms)
10^5
10^4
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
depth: 1
2
3
4
5
6


Vertex Point Subdivision
OpenSubdiv
Ours
timings (ms)
10^0
10^{-1}
10^{-2}
depth: 1
2
3
4
5
6


Our Kernel Timings
ClearBuffer
FacePoints
EdgePoints
VertexPoints
Halfedges
Creases
timings (ms)
10^0
10^{-1}
10^{-2}
depth: 1
2
3
4
5
6

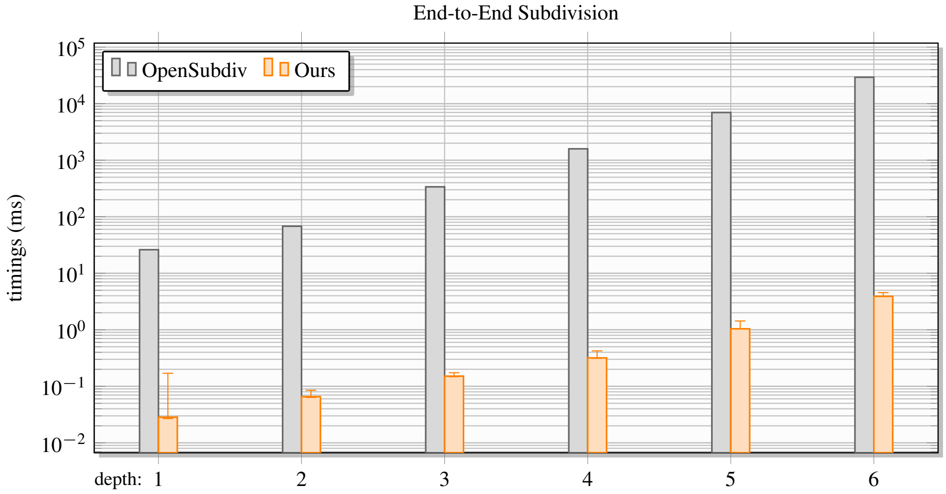
2. Bishop

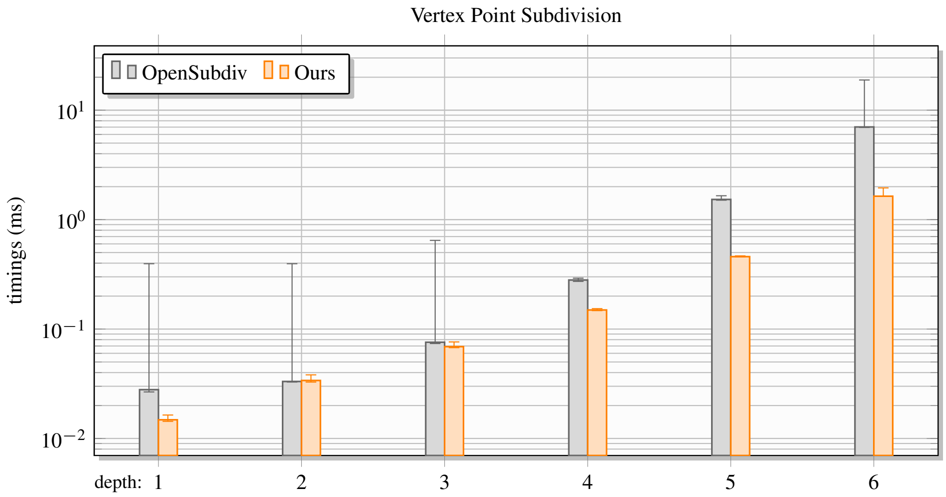
Bishop

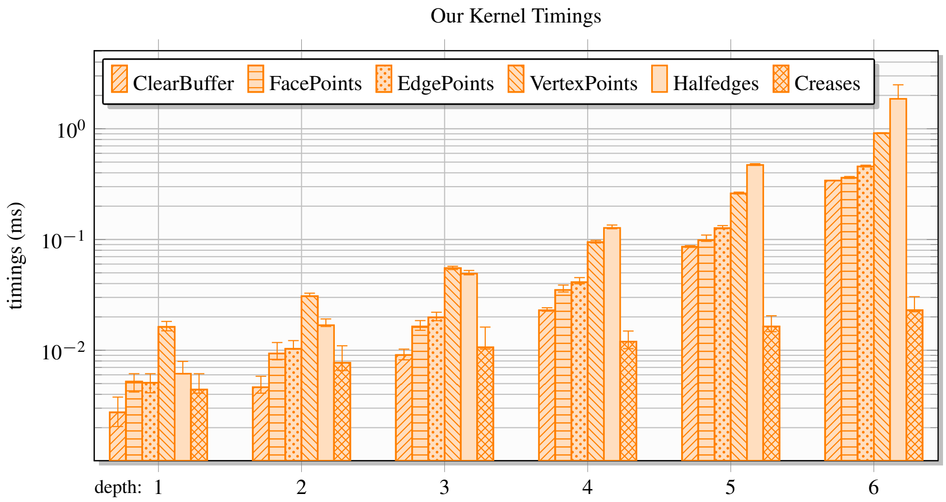


(132 non-quads / 24 boundaries
224 creases)

$H_0 = 3,740$	$H_4 = 957,440$
$F_0 = 968$	$F_4 = 239,360$
$E_0 = 1,882$	$E_4 = 478,912$
$V_0 = 917$	$V_4 = 239,555$

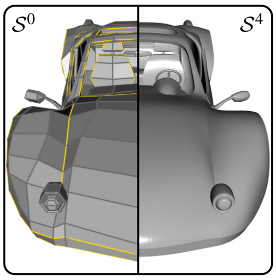






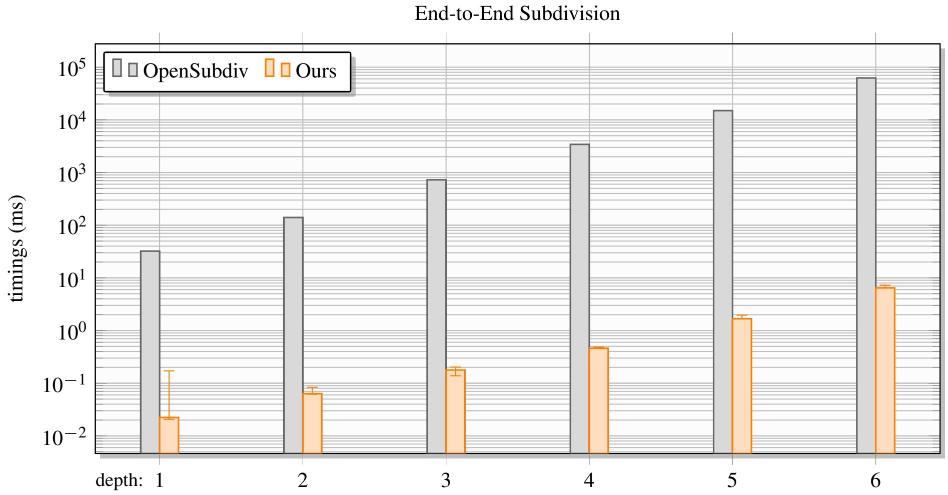
3. Car

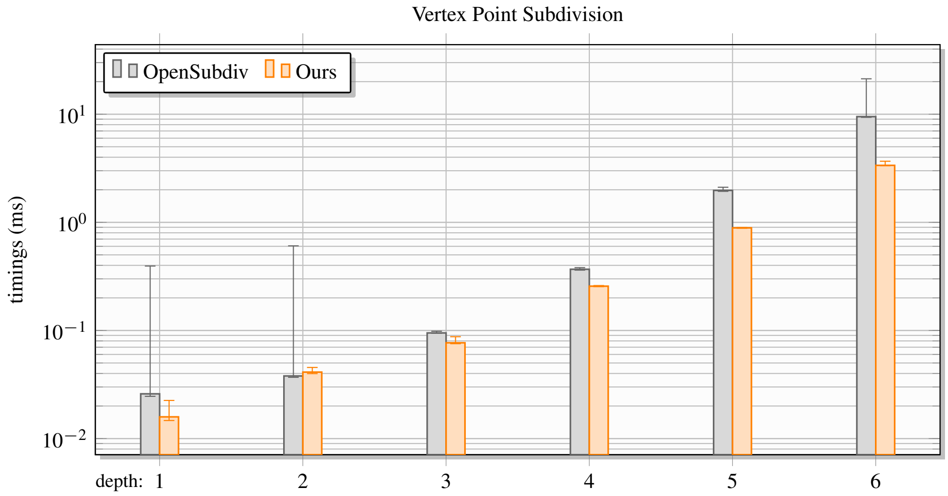
Car

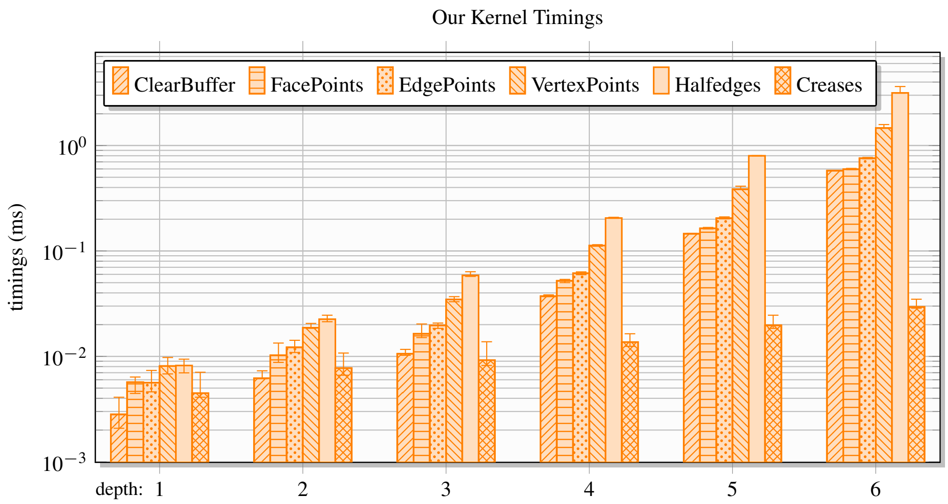


(all-quads / 60 boundaries
314 creases)

$H_0 = 6,300$ $H_4 = 1,612,800$
$F_0 = 1,575$ $F_4 = 403,200$
$E_0 = 3,180$ $E_4 = 806,880$
$V_0 = 1,642$ $V_4 = 403,717$

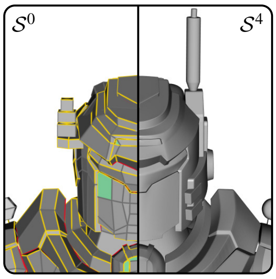






4. ArmorGuy

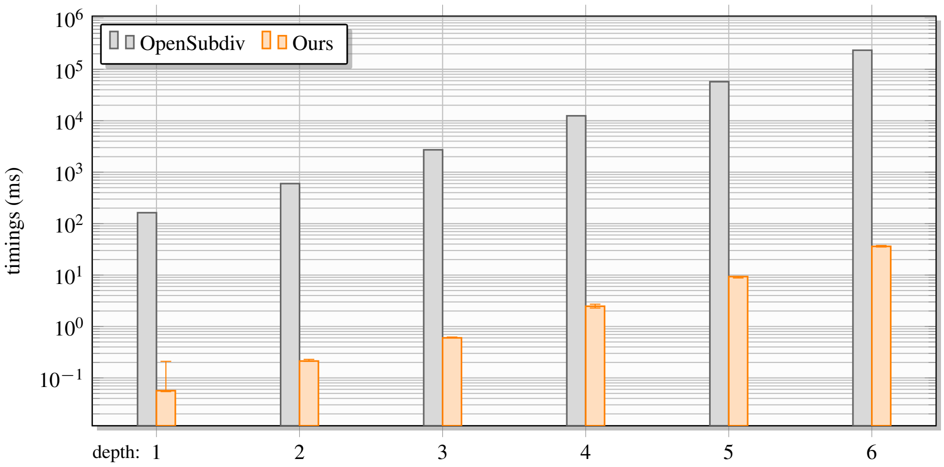
ArmorGuy



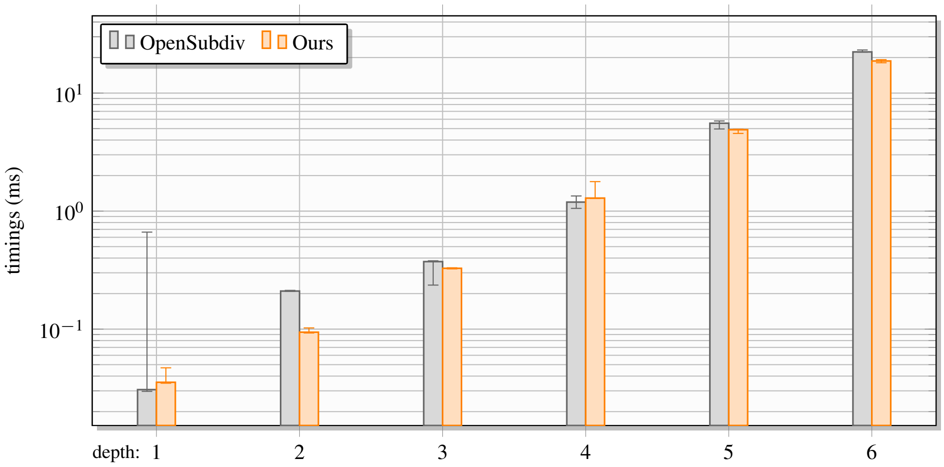
(300 non-quads / 2,034 boundaries
7,101 creases)

$H_0 = 34,388$ $H_4 = 8,803,328$
$F_0 = 8,639$ $F_4 = 2,200,832$
$E_0 = 18,211$ $E_4 = 4,417,936$
$V_0 = 10,022$ $V_4 = 2,217,554$

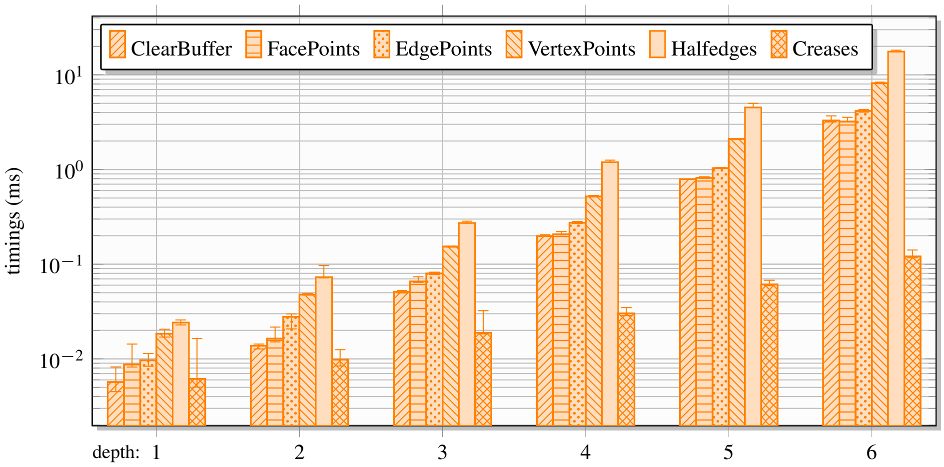
End-to-End Subdivision



Vertex Point Subdivision

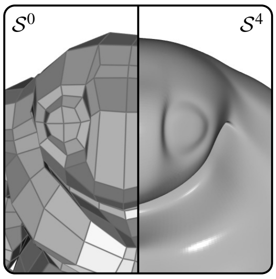


Our Kernel Timings



5. Bigguy

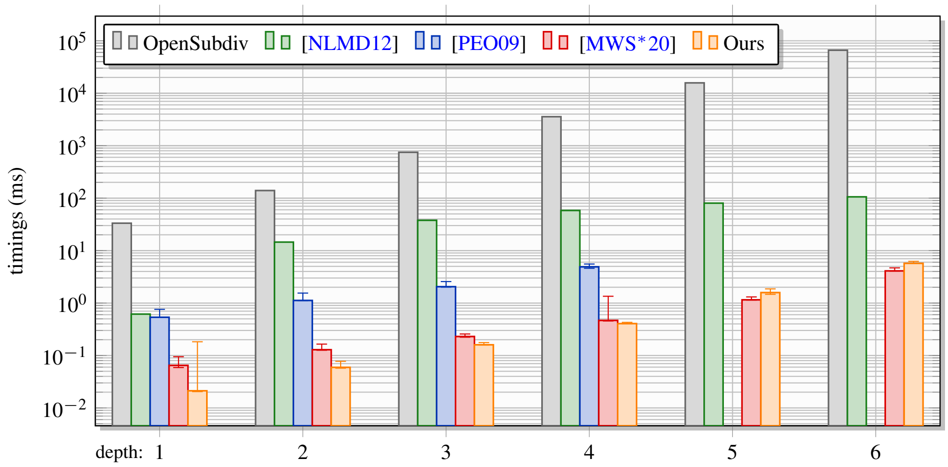
Bigguy

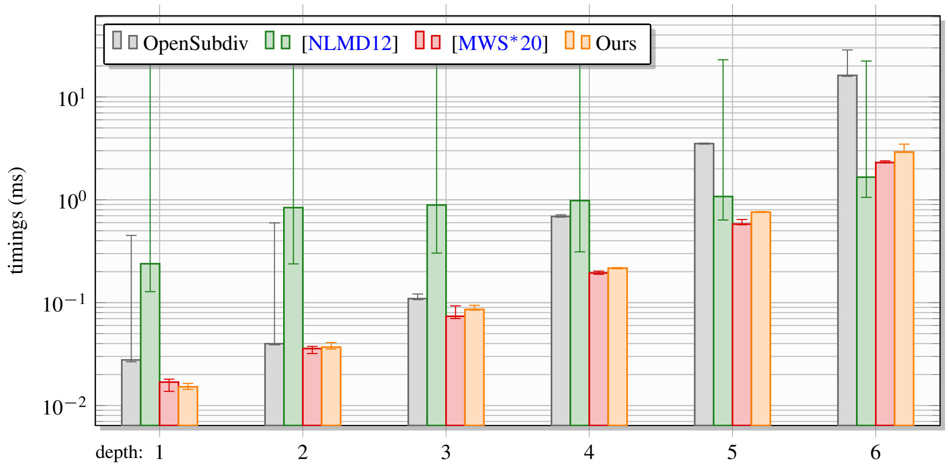


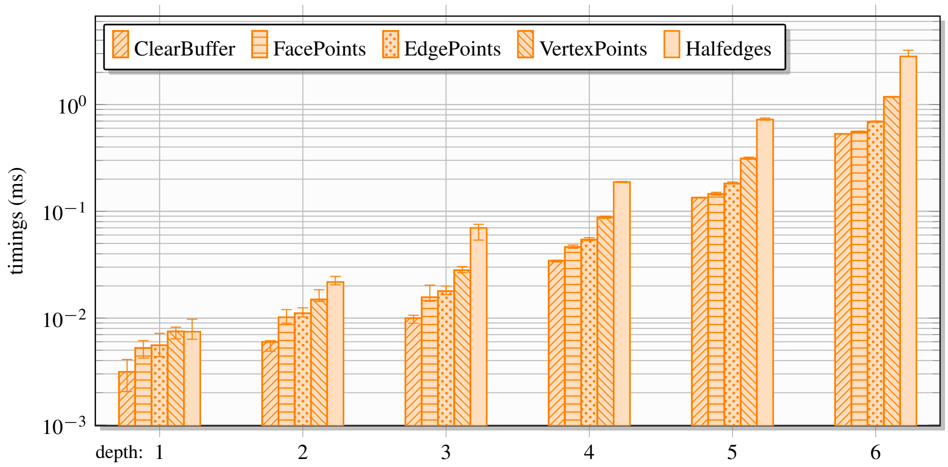
(all-quads / no boundaries)

$H_0 = 5,800$ $F_0 = 1,450$ $E_0 = 2,900$ $V_0 = 1,452$

$H_4 = 1,484,800$ $F_4 = 371,200$ $E_4 = 742,400$ $V_4 = 371,202$

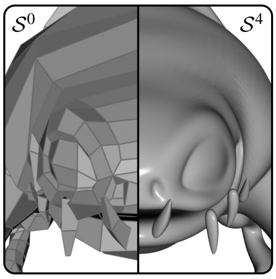






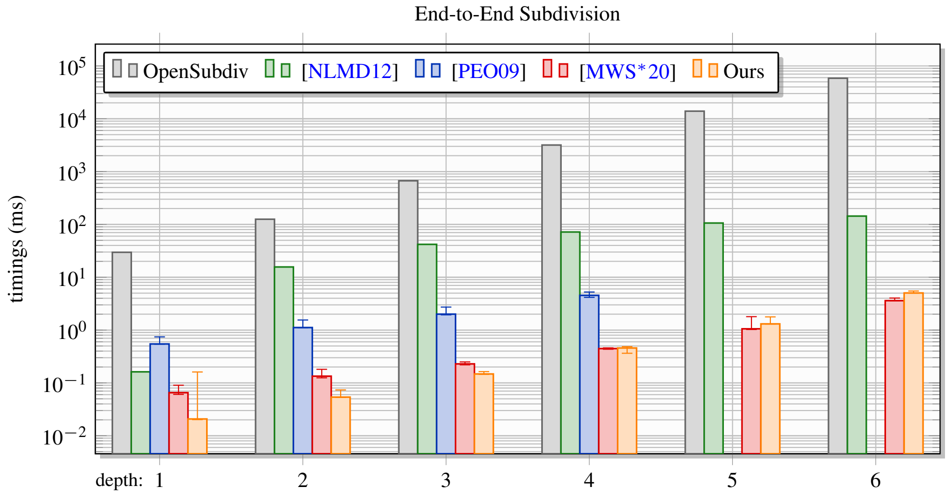
6. Monsterfrog

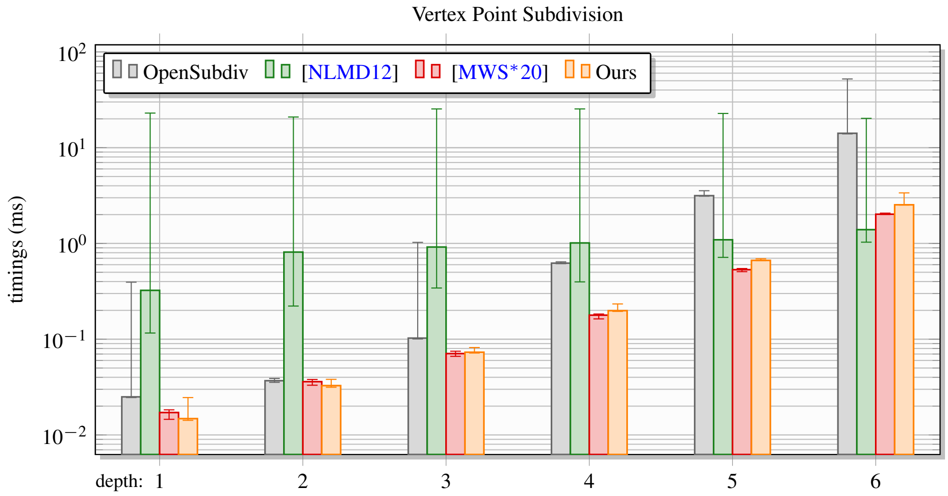
Monsterfrog

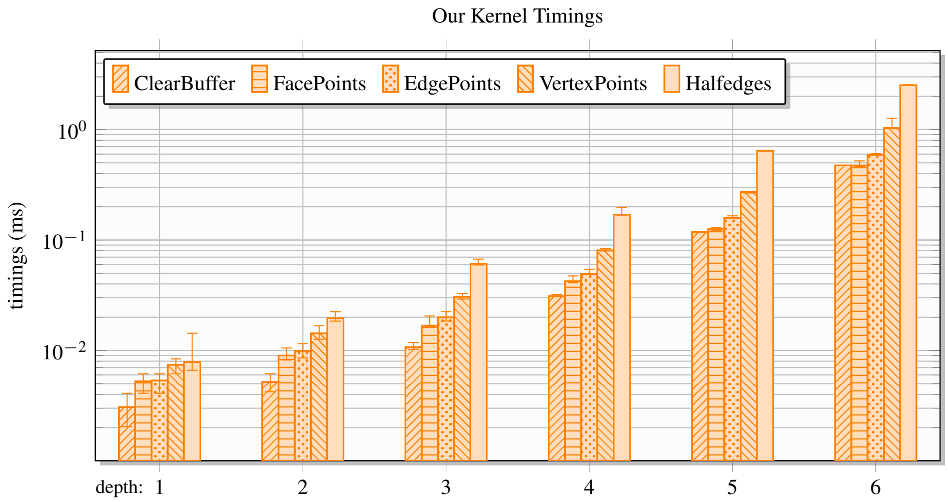


(all-quads / no boundaries)

$H_0 = 5,168$ $H_4 = 1,323,008$
$F_0 = 1,292$ $F_4 = 330,752$
$E_0 = 2,584$ $E_4 = 661,504$
$V_0 = 1,308$ $V_4 = 330,768$

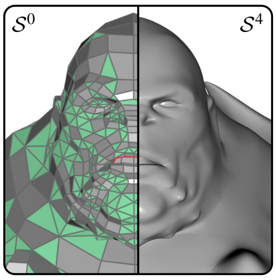






7. Imrod

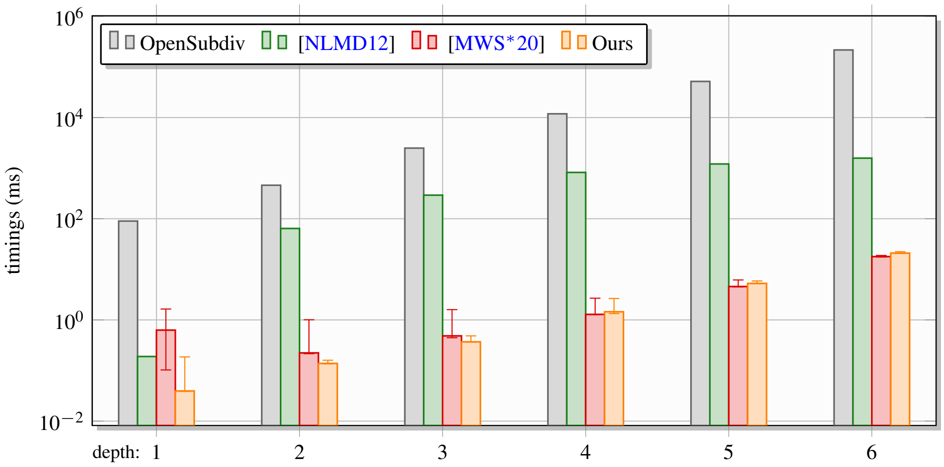
Imrod



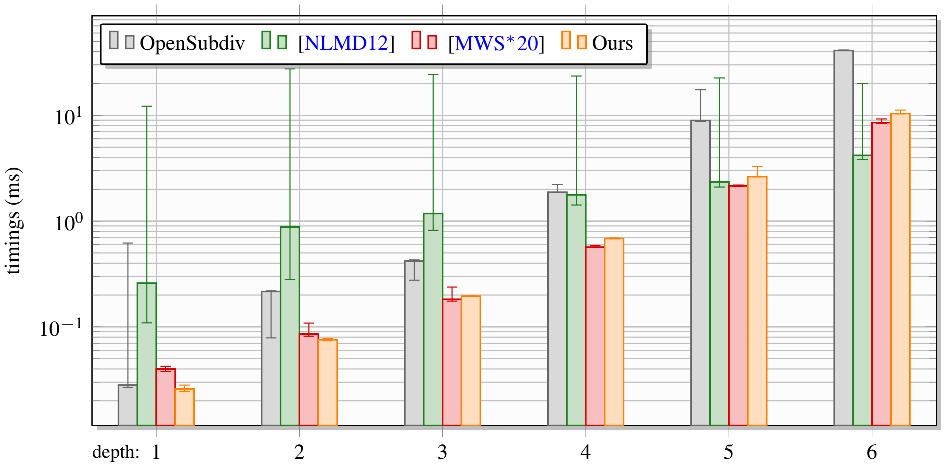
(3,479 non-quads / 223 boundaries)

$H_0 = 21,399$ $H_4 = 5,478,144$
$F_0 = 6,202$ $F_4 = 1,369,536$
$E_0 = 10,811$ $E_4 = 2,740,856$
$V_0 = 4,630$ $V_4 = 1,371,341$

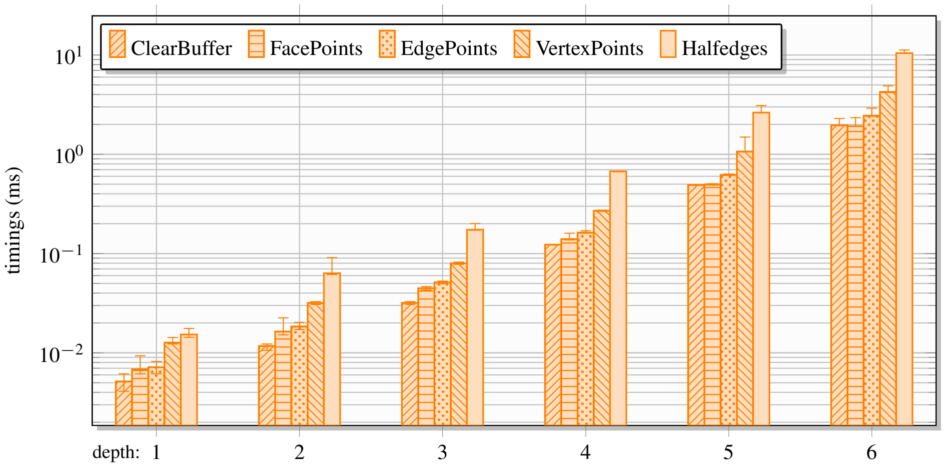
End-to-End Subdivision



Vertex Point Subdivision

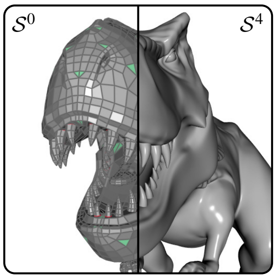


Our Kernel Timings



8. T-Rex

T-Rex



(468 non-quads / 594 boundaries)

$H_0 = 45,224$	$H_4 = 11,577,344$
$F_0 = 11,422$	$F_4 = 2,894,336$
$E_0 = 22,909$	$E_4 = 5,793,424$
$V_0 = 11,539$	$V_4 = 2,899,140$