Universitatea Babeş-Bolyai
Cluj-Napoca

Facultatea de Matematică și Informatică
Departamentul de Informatică

High-Performance Ray Tracing on Modern Parallel Processors

Rezumatul tezei de doctorat

Autor:
Attila T. Áfra

Conducător științific:
Prof. Dr. Horia F. Pop

Cluj-Napoca
2013

Lista publicațiilor

Articole de jurnal ISI

ÁFRA A. T., SZIRMAY-KALOS L.: Stackless Multi-BVH traversal for CPU, MIC and GPU ray tracing. *Computer Graphics Forum* (2013). Acceptat spre publicare.

ÁFRA A. T.: Interactive ray tracing of large models using voxel hierarchies. *Computer Graphics Forum 31*, 1 (2012), 75–88. Prezentat la *Eurographics 2013* (Girona, Spain, 2013). Citări independente: 2 [KSY13, SW13].

Articole de conferință internațională

ÁFRA A. T.: Incoherent ray tracing without acceleration structures. In *Eurographics 2012 - Short Papers* (Cagliari, Sardinia, Italy, 2012), Eurographics Association, pp. 97–100. Citări independente: 3 [NIDN13, VBHH13, CPJ13].

ÁFRA A. T.: Improving BVH ray tracing speed using the AVX instruction set. In *Eurographics 2011 - Posters* (Llandudno, UK, 2011), Eurographics Association, pp. 27–28.

Raporturi tehnice

ÁFRA A. T.: *Faster Incoherent Ray Traversal Using 8-Wide AVX Instructions*. Tech. rep., Babeş-Bolyai University, Cluj-Napoca, Romania, Aug. 2013.

Granturi de cercetare

Grantul **OTKA K-104476** al Fondului Maghiar de Cercetare Științifică: *Physics Simulation and Inverse Problem Solution on Massively Parallel Systems*.

Cuprins

Cuprinsul tezei

Rezumat

Cuvinte cheie: grafică pe calculator, sinteză de imagine realistică, ray tracing, algoritmi paraleli

1 Introducere

Una dintre cele mai fundamentale probleme în grafica pe calculator este de a genera imagini realistice sau stilizate de scene virtuale tridimensionale. Acest proces este numit *randare* sau *sinteză de imagine*. Randarea are numeroase aplicații importante într-o mare varietate de domenii (de exemplu, CAD, arhitectură, vizualizare medicală, filme, jocuri).

În multe cazuri, imaginile randate trebuie să fie atât de fotorealistice cât este posibil. *Ray tracing* [Gla89, SM03] este un algoritm de randare puternic și elegant, care realizează acest lucru prin simularea interacțiunilor de raze de lumină cu obiectele din scenă (vezi Figura 1).

Ray tracing este un algoritm *paralel* în mod inerent pentru că razele de lumină pot fi urmărite independent una de cealaltă. Aceasta este o proprietate foarte utilă, deoarece procesoarele sunt din ce în ce mai mult paralele, dar exploatarea deplină a paralelismului disponibil este o problemă dificilă. O altă problemă majoră este cantitatea de memorie necesară pentru a randa o imagine.

În această teză, prezentăm o colecție de algoritmi de ray tracing noi de înaltă performanță, care abordează problemele menționate mai sus. Acești algoritmi îmbunătățesc eficiența de calcul și reduc cerințele de memorie a metodelor avansate de ray tracing pe arhitecturi de procesoare paralele moderne (CPU, MIC și GPU).

Ray tracing

Algoritmul ray tracing generează imagini prin construirea unor *căi de transport lumină*, care conectează pixeli pe planul imaginii cu surse de lumină în scena virtuală. Fundamentul sintezei de imagine bazată fizic este *ecuația de randare* [Kaj86, ICG86]. Rezolvarea ecuației de randare este de obicei făcută cu metode *Monte Carlo ray tracing* [Szi08] (de exemplu, *path tracing* [Kaj86]).

Figura 1: Exemplu pentru o imagine fotorealistică randată cu ray tracing. *Sursă: "Greek Vases" de Florin Mocanu.*

O operațiune fundamentală în ray tracing este *ray shooting*, al cărui obiectiv este de a găsi cea mai apropiată intersecție a unei raze cu scena. Eficiența algoritmului ray shooting este unul dintre factorii cheie, care determină performanța generală al unui randator bazat pe ray tracing. Astfel, am ales ray shooting-ul ca subiect central al cercetării noastre.

Procesoare paralele moderne

Majoritatea arhitecturilor de procesoare moderne sunt extrem de paralele și sunt capabili de a exploata paralelismul în aplicații la mai multe niveluri. În această teză, ne concentrăm pe trei arhitecturi de procesoare noi: Intel Sandy/Ivy Bridge [Int12a] (CPU), Intel Knights Corner [Int13b, Int12b] (MIC) și NVIDIA Kepler GK110 [Nvi12b, Nvi12a] (GPU).

Contribuții ale tezei

Teza are următoarele contribuții principale:

1. Setul de instrucțiuni AVX introdus cu arhitectura Intel Sandy Bridge a dublat puterea de vârf de procesare floating-point a CPU-urilor x86. Cu toate acestea, utilizarea eficientă a unităților SIMD 8-wide pentru ray tracing este o problemă dificilă. Ne propunem algoritmi *ray traversal* optimizați AVX pentru raze coerente și incoerente, care asigură o performanță mai mare decât abordările state-of-the-art bazate pe SSE. Folosim BVH-uri binare și 8-way ca structuri de accelerare. Am măsurat îmbunătățiri de până la 74% pentru raze coerente și până la 25% pentru raze incoerente. [Áfr11, Áfr13]

2. Extragerea coerenței ascunse din distribuții de raze random necesită procesarea unor loturi foarte mari de raze [PKGH97, ENSB13]. Cei mai mulți algoritmi ray traversal folosesc o stivă, care crește prohibitiv cerințele de memorie a urmăririi a multor raze

în paralel. În consecință, mărimea loturilor de raze trebuie să fie relativ mică, ceea ce conduce la coerență, și astfel și performanță, suboptimală. Soluția este de a utiliza abordări *fără stivă* (*stackless*), însă pentru unele structuri de accelerare eficiente, cum ar fi *multi bounding volume hierarchy* (Multi-BVH sau MBVH), nici un astfel de algoritm nu a fost propus până acum. Prezentăm un algoritm de traversare fără stivă pentru MBVH-uri 4-way și binare, care înlocuiește stiva de traversare obișnuită cu un mic *bitstack*. Bitstack-ul codifică starea de traversare pentru fiecare nivel folosind *coduri skip*. Acesta reduce dimensiunea totală a stării de traversare de aproximativ 22-51×. Demonstrăm că abordarea noastră are un overhead de calcul scăzut (9-31%) pe ultimele arhitecturi CPU, MIC și GPU. [ÁS13]

3. Randarea modelelor masive formate din sute de milioane sau chiar miliarde de primitive are multe provocări. Astfel de scene, de obicei, depășesc dimensiunea memoriei disponibile, în cazul în care trebuie să utilizăm metode de randare speciale *out-of-core* [YGKM08]. Din păcate, astfel de metode de obicei au limitări severe în interactivitate, calitate vizuală, și complexitate de shading. Propunem o metodă nouă de randare out-of-core bazată pe ray tracing, care suportă umbre și iluminare indirectă precise, și rulează la viteze interactive pe CPU-uri multi-core. Componentele sale cheie sunt: o reprezentare ierarhică de model out-of-core, care stochează triunghiuri și voxeli *level-of-detail* (LOD), o metodă eficientă de gestionare a memoriei cu I/O asincron, și un algoritm de traversare pentru kd-tree-uri bazate pe LOD, care este adecvat pentru mai multe tipuri de raze. Metoda noastră de randare are un set unic de caracteristici, combinând avantajele abordărilor anterioare state-of-the-art. [Áfr12b]

4. Metodele ray tracing standarde construiesc o structură de accelerare pentru scena randată, care trebuie să fie actualizată sau reconstruită de fiecare dată când se modifică geometria. Acest lucru face randarea scenelor dinamice cu ray tracing mult mai dificilă decât cu rasterizare. O abordare recent introdusă numită *divide-and-conquer (DAC) ray tracing* [KW11] elimină necesitatea de a menține o structură de accelerare, oferind în același timp performanță competitivă. Cu toate acestea, foarte puţine cercetări au fost făcute cu privire la punerea în aplicare a acestei abordări pe arhitecturi paralele. Un alt domeniu neexplorat este profitarea de actuala distribuție de raze pentru a efectua împărțirea primitivelor mai eficient, ceea ce nu este posibil cu structuri de accelerare preconstruite. Introducem un algoritm eficient ray traversal de tip DAC optimizat pentru raze incoerente și procesare SIMD (folosind instrucțiuni SSE și AVX). În plus față de aceste optimizări, ne propunem *partiționare adaptivă* bazată pe raportul de raze active / primitive, care reduce overheadul de partiționare. Demonstrăm că abordarea noastră surclasează state-of-the-art-ul anterioar de până la 2.2×. [Áfr12a]

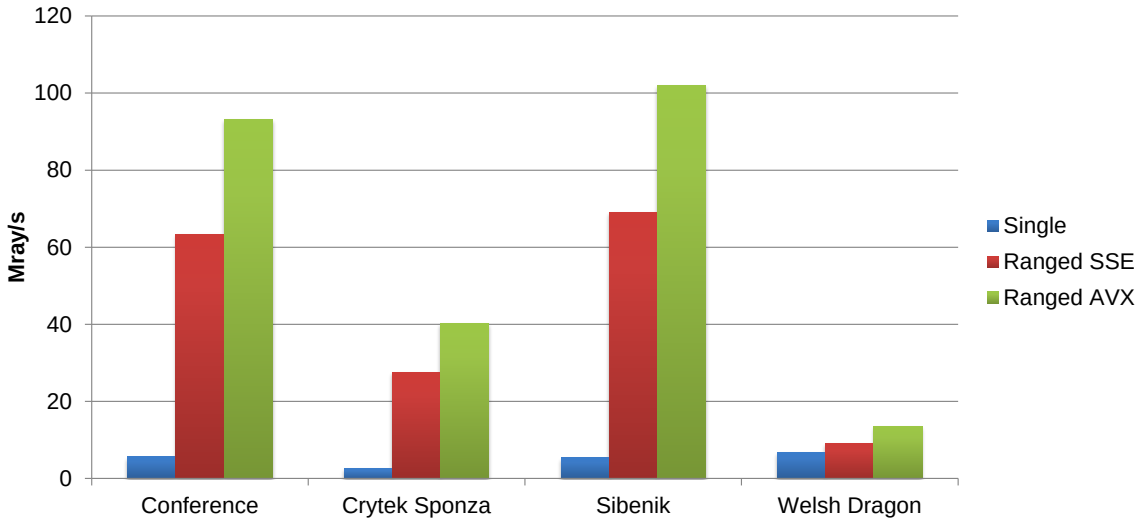


Figura 2: Performanța traversării single și ranged (implementat cu SSE și AVX) pentru raze primare în Mray/s.

2 Traversare coerentă ray packet cu setul de instrucțiuni AVX

Introducere

Algoritmii *ray packet* (*pachete de raze*) [WSBW01, WBS07, ORM08] permit urmărirea rapidă a razelor *coerente*, ceea ce este deosebit de importantă pentru soluții de ray tracing în timp real. O sursă importantă de îmbunătățire a performanței este utilizarea operațiunilor SIMD furnizate de CPU.

Până de curând, arhitecturile CPU populare, cum ar fi x86, au fost capabile să opereze cu până la 4 numere floating-point în mod simultan. Acest lucru s-a schimbat odată cu introducerea setului de instrucțiuni de 256 biți AVX (Advanced Vector Extensions) [Int13a], care a dublat lățimea SIMD a seturilor SSE, AltiVec, NEON, etc.

Am optimizat doi algoritmi BVH packet traversal pentru AVX [Áfr11]: *ranged traversal* [WBS07] și *partition traversal* [ORM08].

AVX ray packet tracing

Cel mai mic primitiv de rază utilizat în algoritmii ranged și partition traversal este *raza SIMD*, care constă din mai multe raze care sunt urmărite împreună de-a lungul întregului algoritm. În implementări SIMD 4-wide, o rază SIMD conține de obicei 2×2 raze, deci razele din jur sunt împachetate împreună pentru a maximiza coerența. Pentru AVX, folosim raze SIMD de 4×2.

Pachetele de raze pot fi destul de mari, având în mod obișnuit o dimensiune de 256 sau chiar 1024 raze, prin urmare, este important de a stoca datele razelor într-un mod cache-

friendly. Acest lucru poate fi realizat prin utilizarea unur modele *array-of-structures-of-arrays* (AoSoA).

Performanța algoritmului ray packet tracing poate fi îmbunătățită și prin aplicarea unor tehnici *frustum culling*. Noi folosim aritmetică intervalară (IA) [WBS07] pentru selectarea nodurilor, și folosim raze de colț pentru selectarea triunghiurilor, așa cum este descris în [BWS06].

Rezultate

Toate testele au fost efectuate pe un sistem cu un procesor Intel Core i5-2400. Rezoluția de randare a fost setată la 1024×768 pixeli.

Figura 2 afișează rezultatele de performanță pentru raze primare. AVX prevede o accelerare, comparativ cu SSE, de cel puțin aproximativ 50% în majoritatea cazurilor. Această creștere subliniară se datorează razelor mari SIMD cu utilizare mai mică și a părților non-SIMD ale algoritmului.

3 Traversare incoerentă cu setul de instrucțiuni AVX

Introducere

În acest capitol, ne propunem un algoritm de traversare single-ray optimizat AVX pentru structura de accelerare MBVH [Áfr13]. MBVH permite o utilizare înaltă SIMD pentru traversare single-ray, de aceea, aceasta este o alegere bună pentru ray tracing *incoerent*. Abordarea noastră oferă o performanță de ray tracing mai mare decât metodele anterioare bazate pe SSE, pentru o mare varietate de cazuri de testare.

Algoritmul de traversare

În algoritmul nostru, urmărim razele în loturi (de exemplu, 256 raze), dar le urmărim în mod individual. La fel ca în algoritmul de traversare a BVH-urilor binare [WBS07], traversăm MBVH-ul N-way cu traversare ordonată depth-first, care are nevoie de o stivă de traversare. Noi folosim distanțele de intersecție furnizate de algoritmul *box test* pentru a determina ordinea de traversare a copiilor intersectați. O modalitate simplă de a realiza acest lucru este de a face sortare orizontală SIMD [FAN07], care, din păcate, este destul de costisitoare și are eficiență SIMD suboptimală.

Dacă sunt mai puțin de N noduri intersectate, sortarea SIMD este chiar mai puțin eficientă, deoarece aceasta întotdeauna sortează pe N valori. Pentru aproximativ 90% din intersecțiile multi-nod valide, doar 1–3 copii sunt loviți. O singură lovitură este cea mai probabilă posibilitate (40–60%). Prin urmare, sortarea SIMD aproape întotdeauna funcționează la o eficiență foarte scăzută, în special pentru factori de ramificare mari.

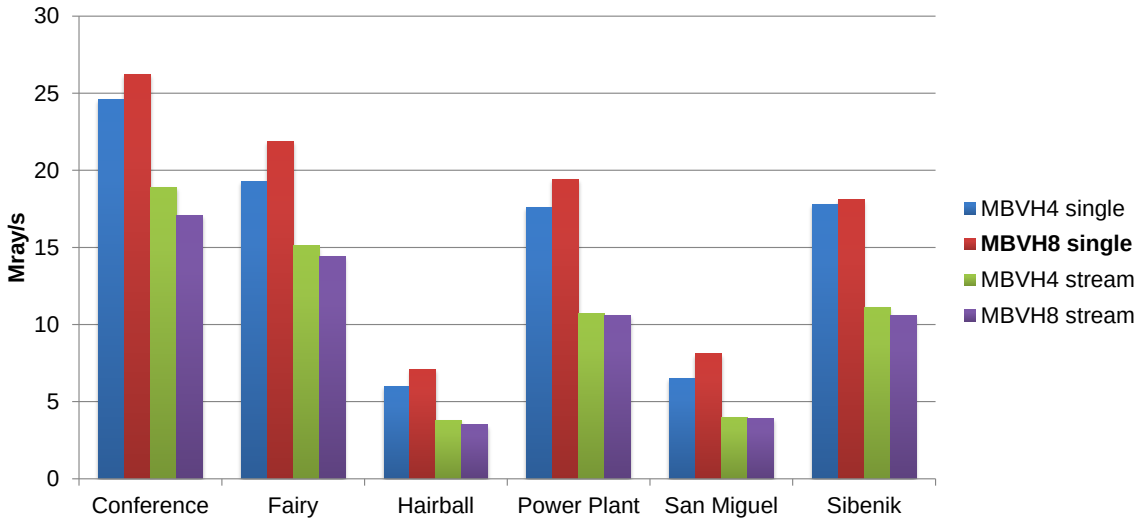


Figura 3: Performanță de traversare pentru raze difuze 8-bounce în Mray/s.

Sortăm nodurile cu o metodă scalară optimizată pentru un număr redus de lovituri, care a fost introdusă în ray tracer-ul Intel Embree pentru MBVH-uri 4-wide [Ern11]. Aceasta este mult mai rapidă decât sortarea SIMD. Ideea principală a metodei este de a folosi implementări specializate de sortare pentru cele mai frecvente numere de lovituri. Am extins algoritmul inițial pentru arbori mai lați prin implementarea următoarelor cazuri: 1, 2, 3, 4 și 5–*N* lovituri.

Rezultate

Am comparat metoda noastră cu traversare MBVH4 single-ray și, de asemenea, cu traversare MBVH RS, care, spre deosebire de alte metode, este capabilă de a extrage coerența ascunsă de raze. Sistemul nostru de referință a avut un procesor Intel Core i7-3770. Am folosit SSE pentru metodele de traversare MBVH4 și AVX pentru cei MBVH8.

Rezultatele de performanță în milione de raze pe secundă pentru raze difuze 8-bounce sunt prezentate în Figura 3. Metoda noastră, traversare MBVH8 single-ray implementat cu AVX, oferă o accelerare de 2–25% față de MBVH4 pentru tipurile de raze și scenele testate. De asemenea, este mai rapidă decât MBVH RS în toate testele noastre, inclusiv benchmarkurile pentru raze primare.

4 Traversare Multi-BVH fără stivă pentru ray tracing pe CPU, MIC și GPU

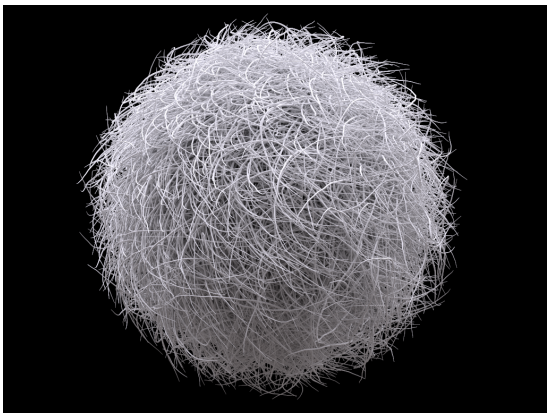
Introducere

Algoritmii de traversare pot fi împărțiți în două categorii principale: algoritmi *cu stivă* și *fără stivă*. Folosirea unei stive pentru traversare este de obicei abordarea cea mai simplă și eficientă.

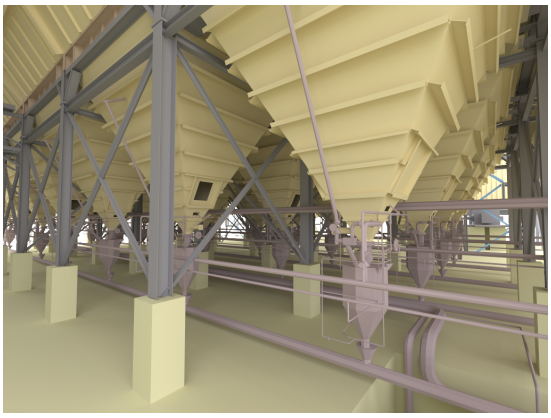
(a) CONFERENCE (282K de triunghiuri)

(b) CRYTEK SPONZA (262K de triunghiuri)

(c) FAIRY (174K de triunghiuri)



(d) HAIRBALL (2.9M de triunghiuri)



(e) POWER PLANT (12.7M de triunghiuri)

(f) SAN MIGUEL (10.5M de triunghiuri)

Figura 4: Scene de test pentru ray traversal fără stivă. Imaginile au fost randate cu path tracing difuz 8-bounce.

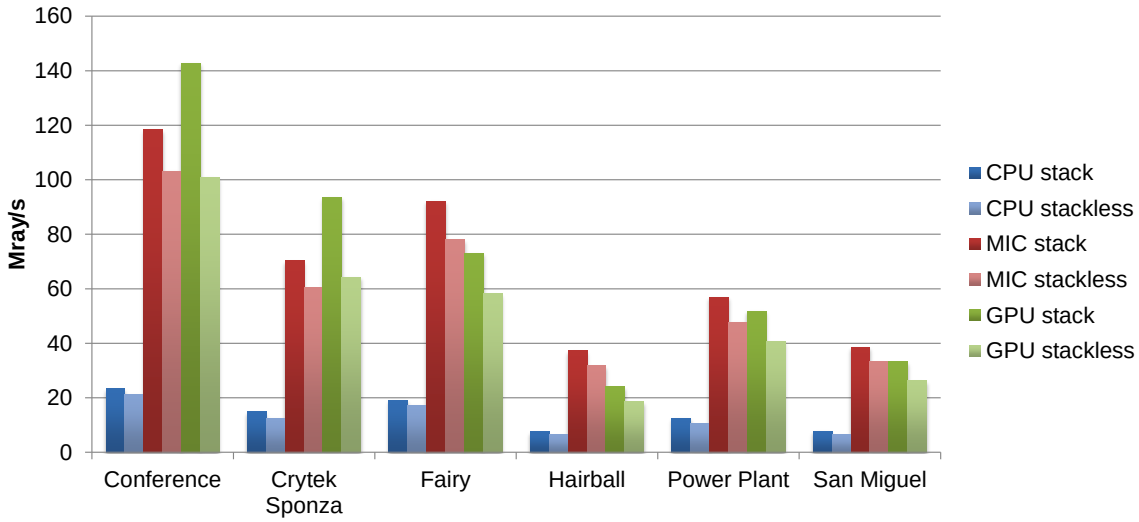


Figura 5: Performanța traversării cu și fără stivă pentru path tracing 8-bounce.

Cu toate acestea, în cazul în care mai multe raze sunt urmărite în paralel, costurile de stocare și de lățime de bandă a menținerii unei stive complete pentru fiecare rază pot fi foarte ridicate.

În acest capitol, ne propunem un nou algoritm eficient ray traversal fără stivă pentru MBVH-uri, care suportă traversare ordonată pe baza distanțelor, fără restartări [ÁS13]. Adăugăm pointeri părinte și frate în arbore fără a crește neapărat amprenta de memorie, și înlocuim stiva obișnuită cu un *bitstack* (*stivă de biți*) compact, un întreg care se încadrează în una sau două registre. În bitstack stocăm *coduri skip*, care indică nodurile frate care trebuie să fie traversate.

Două variante ale algoritmului nostru sunt prezentate: o variație pentru MBVH-uri 4-way (MBVH4) și una pentru BVH-uri binare cu două volume pe nod (MBVH2). MBVH4 este utilizat în principal pe procesoare cu SIMD 4-wide și 8-wide, și, de asemenea, pe arhitectura recentă Intel MIC cu SIMD 16-wide. Pe de altă parte, MBVH2 este alegerea preferată pe GPU-rile curente NVIDIA [AL09, ALK12]. Am optimizat metoda noastră și am evaluat performanța pentru toate aceste platforme hardware.

Prezentare generală a algoritmului

Algoritmul nostru înlocuieste extragerea din stivă în abordările standarde bazate pe stivă cu backtracking în arborele de la nodul curent. Scopul acestei operațiuni este de a găsi următorul nod neprelucrat, care este un frate al nodului curent sau al unor strămoși ai săi. Pentru a putea urca în arbore, adăugăm un pointer părinte pentru fiecare nod. De asemenea, stocăm și pointeri la frați, pentru accesarea acestora într-un mod eficient.

Backtracking-ul este ghidat de un bitmask, care codifică partea arborelui N-way care trebuie să fie traversată. Acesta stochează $N-1$ biți pentru fiecare nivel de arbore vizitat (cu excepția nivelului de rădăcină), și este actualizat în mod similar cu o stivă, dar cu operațiuni

binare. Prin urmare, numim acest bitmask special un *bitstack*. Valorile în bitstack sunt *coduri skip*. Acestea indică frații nodului cel mai recent vizitat pe nivelul respectiv care trebuie omiși.

Rezultate

Am evaluat performanța algorimilor cu și fără stivă cu ajutorul unui path tracer difuz simplu dar optimizat pe următoarele arhitecturi: Intel Core i7-3770 (CPU), Intel Xeon Phi SE10P (MIC) și NVIDIA Tesla K20c (GPU).

Rezultatele de performanță sunt prezentate în Figura 5. Algoritmii noștri fără stivă sunt mai lenți decât cei de referință cu stivă, dacă sunt utilizați pentru ray tracing obișnuit, dar starea lor de traversare este mai mică de aproximativ 22–51×. Pentru scenele noastre de testare (Figura 4), traversarea fără stivă este mai lentă cu 9–17% pe CPU, 13–16% pe MIC și 20–31% pe GPU.

5 Ray tracing interactiv al modelelor mari cu ierarhii voxel

Introducere

Acest capitol prezintă o nouă metodă de randare bazată pe ray tracing pentru modele masive [Áfr12b], care combină eficient avantajele și tehnicile a diferitelor abordări existente.

Mai multe exemple de testare demonstrează că metoda noastră funcționează în mod eficient pentru diferite tipuri de modele complexe, realizând rate de cadre interactive pe un desktop PC quad-core. Metoda suportă o mare varietate de algoritmi de shading bazate pe ray tracing, care includ iluminare directă cu umbre, ocluzie ambientală și iluminație globală (vezi Figura 6).

Prezentare generală a metodei

În primul rând, construim o *structură de date ierarhică out-of-core*, care conține, într-un format comprimat, triunghiurile originale și mai multe niveluri LOD compuse din voxeli.

Datorită mecanismului LOD ierarhic, este posibil să randăm seturi de date imense, care nu pot fi complet încărcate în memoria sistemului. Încărcăm detaliile necesare într-un mod *asincron*, astfel, nu sunt degradări de performanță din cauza datelor insuficiente.

Organizăm toate primitivele (i.e., triunghiuri și voxeli) într-un kd-tree. Acest kd-tree out-of-core are un scop dublu în abordarea noastră: se accelerează intersecțiile razelor cu triunghiuri și stochează ierarhia voxel.

Unele noduri de kd-tree conțin un singur *voxel LOD*, care este un primitiv randat ca un cuboid aliniat pe axe. Acesta aproximează primitivele originale stocate în subarborele nodului corespunzător și deține *atributuri de shading* (de exemplu, vector normal și culoare) pentru fiecare față a cuboidului.

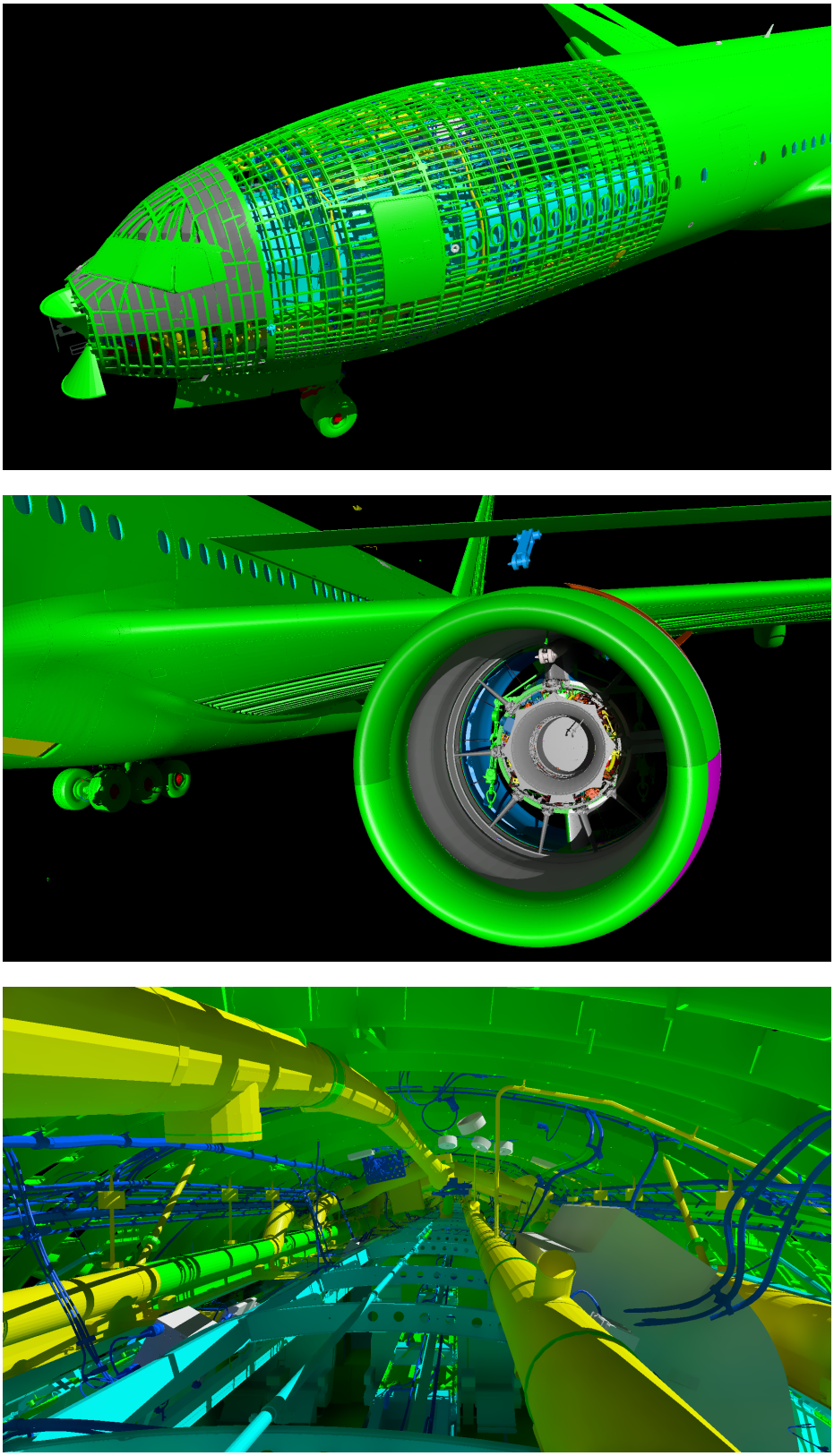


Figura 6: Modelul BOEING 777 randat interactiv cu umbre și iluminație indirectă.

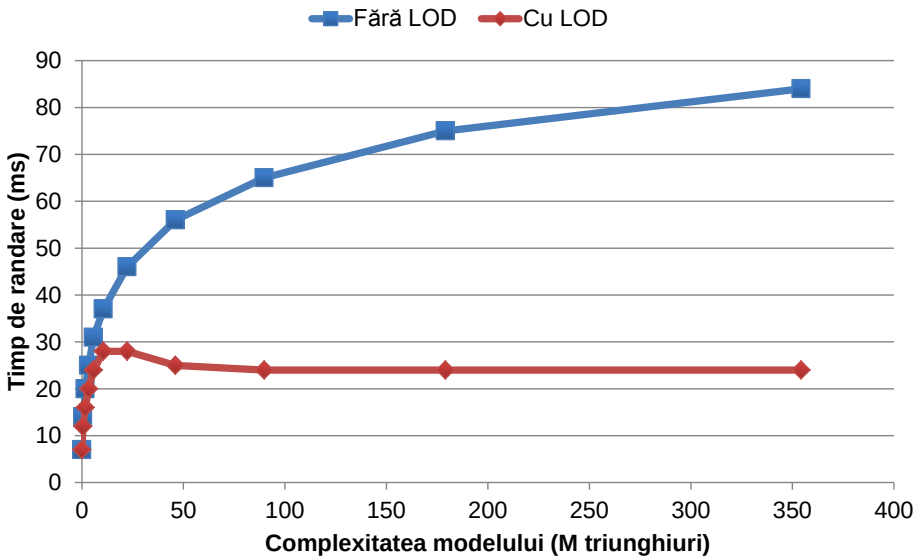


Figura 7: Scalarea performanței de ray casting cu complexitatea modelului pentru MANDELBULB.

Întreagul kd-tree este descompus în *treelet-uri*, care sunt grupate în *blocuri* de mărimi identice. În scopul de a reduce cerințele de stocare, blocurile sunt codificate utilizând un algoritm de compresie a datelor fără pierderi.

Folosim un *operator de gestiune a memoriei* implementat pur în software, care este responsabil pentru încărcarea blocurilor cerute de randator.

Integrarea strânsă a nivelurilor LOD cu structura de accelerare permite o reprezentare de model și un algoritm ray traversal eficient. Prin utilizarea voxelurilor LOD, rata de cadre poate să fie mult mai mare, cu pierderi minime de calitate a imaginii. Noi oferim metrici de eroare LOD rapide pentru raze primare, de umbră, de ocluzie ambientală, și de interreflexiune difuză.

Rezultate

Toate benchmarkurile au fost efectuate pe un desktop PC cu un CPU Intel Core i7-2600, 8 GB de RAM, un GPU NVIDIA GeForce GTX 560 Ti, și două hard diskuri 7200 RPM în RAID 0.

Am ales modele de testare din diferite domenii de aplicare: POWER PLANT (12M de triunghiuri), ASIAN DRAGON (7M de triunghiuri), LUCY (28M de triunghiuri), MPI V1.0 (73M de triunghiuri), BOEING 777 (337M de triunghiuri) și MANDELBULB (354M de triunghiuri).

Scalarea performanței de ray casting cu numărul de triunghiuri este ilustrată în Figura 7. Fără LOD, timpul de randare crește logaritmic, iar dacă utilizăm voxelurile LOD, performanța devine aproape constantă după un anumit punct.

Demonstrăm că chiar și cele mai complexe modele din suita de testare pot fi randate la viteze interactive cu abordarea noastră.

6 Ray tracing incoerent fără structuri de accelerare

Introducere

Un ray tracer de obicei este format din două părți principale: ray traversal și construcția structurii de accelerare. Keller și Wächter [KW11] au propus recent o abordare diferită și elegantă numită *divide-and-conquer ray tracing*, care nu are nevoie de o structură de accelerare.

În acest capitol, ne propunem un nou algoritm de traversare DAC [Áfr12a] bazat pe metoda de Keller et al. Abordarea noastră este în general mai eficientă decât metoda lui Mora [Mor11], și exploatează setul de instrucțiuni AVX. Am optimizat metoda noastră pentru raze incoerente.

Filtrarea razelor

Filtrarea poate fi executată pe loc prin rearanjarea listei de raze, pentru a crea o partiție *activă* și una *inactivă*. O rază este specificată cu un punct de origine, un vector de direcție, un interval $[0, t_{\max}]$ și un ID. Dimensiunea totală a unei raze este de 32 octeți, ceea ce înseamnă că se încadrează într-un singur registru AVX sau două registre SSE.

Evităm problemele de cache prin simpla rearanjare a razelor în lista originală. Razele pot fi copiate într-un mod rapid în blocuri de 32 (cu AVX) sau 16 octeți (cu SSE).

Intersectăm simultan 4 raze când utilizăm SSE și 8 raze atunci când utilizăm AVX. Înainte de a face acest lucru, datele de raze trebuie rearanjate în format SoA.

Partiționarea triunghiurilor

Partiționarea triunghiurilor împarte o listă de triunghiuri în două subliste disjuncte. Noi folosim două metode de partiționare diferite: *partiționare la mijloc* și *partiționare SAH*.

Algoritmii de partiționare nu procesează direct triunghiurile, ci numai AABB-urile (axis-aligned bounding box) lor, care sunt precomputate. În contrast cu filtrarea razelor, gestionăm un tablou cu ID-uri de triunghi în loc de reordonarea directă a AABB-urilor.

Partiționarea cu SAH nu duce neapărat la cea mai mare performanță posibilă de ray tracing. Rezolvăm această problemă prin decizia adaptivă între SAH și partiționare la mijloc. În fiecare etapă de partiționare, raportul dintre numărul de raze active și triunghiuri actuale este comparată cu un prag predefinit (de exemplu, 1–2).

Intersecția triunghiurilor

În metoda noastră folosim o reprezentare specială de triunghi pentru a economisi spațiul de memorie și lățimea de bandă. Similar cu filtrarea razelor, se intersectează raze multiple cu triunghiul curent folosind SIMD.

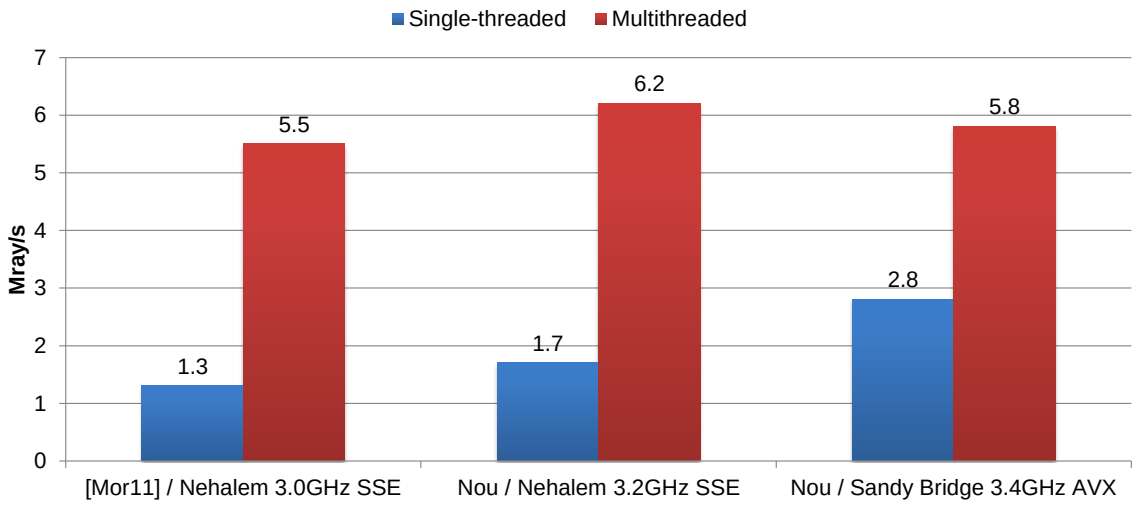


Figura 8: Performanță ray traversal DAC pentru raze difuze 8-bounce și scena CONFERENCE.

Traversare ordonată

Pentru raze primare, traversarea față-la-spate are un impact pozitiv semnificativ asupra vitezei de ray tracing. Cu toate acestea, îmbunătățirea este mică pentru raze incoerente. Determinăm ordinea de traversare cu abordarea foarte ieftină din packet tracer-ul de Wald et al. [WBS07].

Rezultate

Benchmarkurile au fost realizate pe două sisteme diferite: pe un procesor Intel Core i7-960 cu 24 GB de RAM (triple channel) și pe un procesor Intel Core i7-2600 cu 8 GB de RAM (dual channel). Am testat algoritmii folosind un path tracer Monte Carlo cu reflexii difuze.

Metoda noastră este destul de competitivă cu un ray tracer static optimizat care utilizează structura de accelerare MBVH [Ern11]. De exemplu, MBVH este cu doar 12% mai rapid pentru path tracing 8-bounce și scena CONFERENCE, pe un singur fir al procesorului i7-960. Cu toate acestea, diferența este mai mare pe mai multe fire, mai ales pentru HAIRBALL, unde metoda noastră este de 4× mai lentă.

Comparativ cu metoda lui Mora, abordarea noastră filtrează razele mai eficient, folosește partiționare de o calitate mai înaltă, exploatează SIMD mai lat, și este optimizat pentru raze incoerente (vezi Figura 8).

7 Concluzii

În această lucrare, am investigat ray tracing de înaltă performanță pe arhitecturi de procesoare paralele moderne. Mai concret, ne-am propus metode care exploatează mai bine paralelismul și memoria disponibilă pe hardware de ultimă generație, și permit calitate de imagine și interactivitate superioară față de abordările anterioare. Am oferit soluții eficiente pentru randare atât offline, cât și în timp real.

Posibile direcții viitoare de cercetare includ: ray traversal MBVH4 pe GPU, ray tracing out-of-core general, LOD bazat pe voxeli de o calitate superioară, și ray tracing DAC pe procesoare masiv paralele.

Bibliografie

[Áfr10] ÁFRA A. T.: Interactive out-of-core ray casting of massive triangular models with voxel-based LODs. In *Proceedings of the 5th Hungarian Conference on Computer Graphics and Geometry* (Budapest, Hungary, 2010), pp. 4–11.

[Áfr11] ÁFRA A. T.: Improving BVH ray tracing speed using the AVX instruction set. In *Eurographics 2011 - Posters* (Llandudno, UK, 2011), Eurographics Association, pp. 27–28.

[Áfr12a] ÁFRA A. T.: Incoherent ray tracing without acceleration structures. In *Eurographics 2012 - Short Papers* (Cagliari, Sardinia, Italy, 2012), Eurographics Association, pp. 97–100.

[Áfr12b] ÁFRA A. T.: Interactive ray tracing of large models using voxel hierarchies. *Computer Graphics Forum 31*, 1 (2012), 75–88.

[Áfr13] ÁFRA A. T.: *Faster Incoherent Ray Traversal Using 8-Wide AVX Instructions*. Tech. rep., Babeş-Bolyai University, Cluj-Napoca, Romania, Aug. 2013.

[AK90] ARVO J., KIRK D.: Particle transport and image synthesis. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1990), SIGGRAPH '90, ACM, pp. 63–66.

[AK10] AILA T., KARRAS T.: Architecture considerations for tracing incoherent rays. In *Proceedings of the Conference on High Performance Graphics* (Aire-la-Ville, Switzerland, 2010), HPG '10, Eurographics Association, pp. 113–122.

[AL09] AILA T., LAINE S.: Understanding the efficiency of ray traversal on GPUs. In *Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), HPG '09, ACM, pp. 145–149.

[ALK12] AILA T., LAINE S., KARRAS T.: *Understanding the Efficiency of Ray Traversal on GPUs – Kepler and Fermi Addendum*. NVIDIA Technical Report NVR-2012-02, NVIDIA Corporation, June 2012.

[ÁS13] ÁFRA A. T., SZIRMAY-KALOS L.: Stackless Multi-BVH traversal for CPU, MIC and GPU ray tracing. *Computer Graphics Forum* (2013). To appear.

[BA13] BARRINGER R., AKENINE-MÖLLER T.: Dynamic stackless binary tree traversal. *Journal of Computer Graphics Techniques (JCGT) 2*, 2 (March 2013), 38–49.

[Bat68] BATCHER K. E.: Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference* (New York, NY, USA, 1968), AFIPS '68 (Spring), ACM, pp. 307–314.

[BBS*09] BUDGE B., BERNARDIN T., STUART J. A., SENGUPTA S., JOY K. I., OWENS J. D.: Out-of-core data management for path tracing on hybrid resources. *Computer Graphics Forum 28*, 2 (2009), 385–396.

[Ben06] BENTHIN C.: *Realtime Ray Tracing on Current CPU Architectures*. PhD thesis, Computer Graphics Group, Saarland University, 2006.

[Bik12] BIKKER J.: *Ray Tracing in Real-Time Games*. PhD thesis, Delft University of Technology, 2012.

[BWB08] BOULOS S., WALD I., BENTHIN C.: Adaptive ray packet reordering. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing 2008* (Los Alamitos, CA, USA, 2008), IEEE Computer Society, pp. 131–138.

[BWS06] BOULOS S., WALD I., SHIRLEY P.: *Geometric and Arithmetic Culling Methods for Entire Ray Packets*. Tech. Rep. UUCS-06-010, School of Computing, University of Utah, 2006.

[BWW*12] BENTHIN C., WALD I., WOOP S., ERNST M., MARK W.: Combining single and packet-ray tracing for arbitrary ray distributions on the Intel MIC architecture. *IEEE Transactions on Visualization and Computer Graphics 18*, 9 (September 2012), 1438–1448.

[CCC87] COOK R. L., CARPENTER L., CATMULL E.: The Reyes image rendering architecture. *SIGGRAPH '87 21*, 4 (Aug. 1987), 95–102.

[CKL*10] CHOI B., KOMURAVELLI R., LU V., SUNG H., BOCCHINO R. L., ADVE S. V., HART J. C.: Parallel SAH k-D tree construction. In *Proceedings of the Conference on High Performance Graphics* (Aire-la-Ville, Switzerland, 2010), HPG '10, Eurographics Association, pp. 77–86.

[CNLE09] CRASSIN C., NEYRET F., LEFEBVRE S., EISEMANN E.: GigaVoxels: Ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2009), I3D '09, ACM, pp. 15–22.

[CPJ13] COSTA V., PEREIRA J. M., JORGE J. A.: Compressed grids for GPU ray tracing of large models. In *WSCG 2013 - Poster Proceedings* (Plzen, Czech Republic, 2013), Vaclav Skala - Union Agency, pp. 29–32.

[Dam11] DAMMERTZ H.: *Acceleration Methods for Ray Tracing based Global Illumination*. PhD thesis, Ulm University, 2011.

[DHK08] DAMMERTZ H., HANIKA J., KELLER A.: Shallow bounding volume hierarchies for fast SIMD ray tracing of incoherent rays. *Computer Graphics Forum 27*, 4 (2008), 1225–1233.

[EG08] ERNST M., GREINER G.: Multi bounding volume hierarchies. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing 2008* (2008), pp. 35–40.

[ENSB13] EISENACHER C., NICHOLS G., SELLE A., BURLEY B.: Sorted deferred shading for production path tracing. *Computer Graphics Forum 32*, 4 (2013), 125–132.

[Ern11] ERNST M.: Embree: Photo-realistic ray tracing kernels. *SIGGRAPH 2011 Talk* (2011).

[FAN07] FURTAK T., AMARAL J. N., NIEWIADOMSKI R.: Using SIMD registers and instructions to enable instruction-level parallelism in sorting algorithms. In *Proceedings of the 19th Annual ACM Symposium on Parallel Algorithms and Architectures* (New York, NY, USA, 2007), SPAA '07, ACM, pp. 348–357.

[Fra04] FRASER K.: *Practical lock-freedom*. Tech. Rep. UCAM-CL-TR-579, University of Cambridge, Computer Laboratory, Feb. 2004.

[FS88] FUSSELL D. S., SUBRAMANIAN K. R.: *Fast Ray Tracing Using K-d Trees*. Tech. rep., University of Texas, Austin, TX, USA, 1988.

[FS05] FOLEY T., SUGERMAN J.: KD-tree acceleration structures for a GPU raytracer. In *Proceedings of the ACM SIGGRAPH/Eurographics Conference on Graphics Hardware* (New York, NY, USA, 2005), HWWS '05, ACM, pp. 15–22.

[FTI86] FUJIMOTO A., TANAKA T., IWATA K.: ARTS: Accelerated ray-tracing system. *Computer Graphics and Applications, IEEE 6*, 4 (1986), 16–26.

[GKDS12] GEORGIEV I., KŘIVÁNEK J., DAVIDOVIČ T., SLUSALLEK P.: Light transport simulation with vertex connection and merging. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 192:1–192:10.

[Gla89] GLASSNER A. S. (Ed.): *An Introduction to Ray Tracing*. Academic Press Ltd., London, UK, 1989.

[GM04] GOBBETTI E., MARTON F.: Layered point clouds: a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models. *Computers & Graphics 28*, 6 (2004), 815–826.

[GM05] GOBBETTI E., MARTON F.: Far Voxels – a multiresolution framework for interactive rendering of huge complex 3D models on commodity graphics platforms. *ACM Transactions on Graphics 24*, 3 (August 2005), 878–885. Proc. SIGGRAPH 2005.

[GPM11] GARANZHA K., PANTALEONI J., MCALLISTER D.: Simpler and faster HLBVH with work queues. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics* (New York, NY, USA, 2011), HPG '11, ACM, pp. 59–64.

[GR08] GRIBBLE C. P., RAMANI K.: Coherent ray tracing via stream filtering. In *2008 IEEE/Eurographics Symposium on Interactive Ray Tracing* (August 2008), pp. 59–66.

[GS87] GOLDSMITH J., SALMON J.: Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications 7*, 5 (May 1987), 14–20.

[Hav00] HAVRAN V.: *Heuristic Ray Shooting Algorithms*. PhD thesis, Czech Technical University in Prague, November 2000.

[HBŽ98] HAVRAN V., BITTNER J., ŽÁRA J.: Ray tracing with rope trees. In *Proceedings of SCCG'98 (Spring Conference on Computer Graphics)* (Budmerice, Slovak Republic, Apr. 1998), pp. 130–139.

[HDW*11] HAPALA M., DAVIDOVIČ T., WALD I., HAVRAN V., SLUSALLEK P.: Efficient stack-less BVH traversal for ray tracing. In *Proceedings of the 27th Spring Conference on Computer Graphics* (New York, NY, USA, 2011), SCCG '11, ACM, pp. 7–12.

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. In *ACM SIGGRAPH Asia 2008 papers* (New York, NY, USA, 2008), SIGGRAPH Asia '08, ACM, pp. 130:1–130:8.

[HP11] HENNESSY J. L., PATTERSON D. A.: *Computer Architecture: A Quantitative Approach*, 5th ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[HPJ12] HACHISUKA T., PANTALEONI J., JENSEN H. W.: A path space extension for robust light transport simulation. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 191:1–191:10.

[HSHH07] HORN D. R., SUGERMAN J., HOUSTON M., HANRAHAN P.: Interactive k-D tree GPU raytracing. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2007), I3D '07, ACM, pp. 167–174.

[HZDS09] HAVRAN V., ZAJAC J., DRAHOKOUPIL J., SEIDEL H.-P.: *MPI Informatics Building Model as Data for Your Research*. Research Report MPI-I-2009-4-004, MPI Informatik, Saarbruecken, Germany, December 2009.

[ICG86] IMMEL D. S., COHEN M. F., GREENBERG D. P.: A radiosity method for non-diffuse environments. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, ACM, pp. 133–142.

[Int12a] INTEL: *Intel 64 and IA-32 Architectures Optimization Reference Manual*, April 2012.

[Int12b] INTEL: *Knights Corner Instruction Set Reference Manual*, August 2012.

[Int13a] INTEL: *Intel 64 and IA-32 Architectures Software Developer's Manual*, June 2013.

[Int13b] INTEL: *Intel Xeon Phi System Software Developer's Guide*, June 2013.

[Jen96] JENSEN H. W.: Global illumination using photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96* (London, UK, 1996), Springer-Verlag, pp. 21–30.

[Jen01] JENSEN H. W.: *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[JR13] JEFFERS J., REINDERS J.: *Intel Xeon Phi Coprocessor High Performance Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2013.

[KA13] KARRAS T., AILA T.: Fast parallel construction of high-quality bounding volume hierarchies. In *Proceedings of the 5th High-Performance Graphics Conference* (New York, NY, USA, 2013), HPG '13, ACM, pp. 89–99.

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, ACM, pp. 143–150.

[KH01] KELLER A., HEIDRICH W.: Interleaved sampling. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques* (London, UK, UK, 2001), Springer-Verlag, pp. 269–276.

[KIS*12] KOPTA D., IZE T., SPJUT J., BRUNVAND E., DAVIS A., KENSLER A.: Fast, effective BVH updates for animated scenes. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2012), I3D '12, ACM, pp. 197–204.

[KK86] KAY T. L., KAJIYA J. T.: Ray tracing complex scenes. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, ACM, pp. 269–278.

[KSAC02] KELEMEN C., SZIRMAY-KALOS L., ANTAL G., CSONKA F.: A simple and robust mutation strategy for the Metropolis light transport algorithm. *Computer Graphics Forum 21*, 3 (2002), 531–540.

[KSS*13] KOPTA D., SHKURKO K., SPJUT J., BRUNVAND E., DAVIS A.: An energy and bandwidth efficient ray tracing architecture. In *Proceedings of the 5th High-Performance Graphics Conference* (New York, NY, USA, 2013), HPG '13, ACM, pp. 121–128.

[KSY13] KIM T.-J., SUN X., YOON S.-E.: T-ReX: Interactive global illumination of massive models on heterogeneous computing resources. *IEEE Transactions on Visualization and Computer Graphics* (2013). To appear.

[KW11] KELLER A., WÄCHTER C.: Efficient ray tracing without auxiliary acceleration data structure. *High-Performance Graphics 2011 (Poster)* (2011).

[KZ11] KNAUS C., ZWICKER M.: Progressive photon mapping: A probabilistic approach. *ACM Transactions on Graphics 30*, 3 (May 2011), 25:1–25:13.

[Laf96] LAFORTUNE E.: *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering*. PhD thesis, Katholieke Universiteit Leuven, 1996.

[Lai10] LAINE S.: Restart trail for stackless BVH traversal. In *Proceedings of the Conference on High Performance Graphics* (Aire-la-Ville, Switzerland, 2010), HPG '10, Eurographics Association, pp. 107–111.

[LK10] LAINE S., KARRAS T.: Efficient sparse voxel octrees. In *Proceedings of ACM SIGGRAPH 2010 Symposium on Interactive 3D Graphics and Games* (2010), ACM Press, pp. 55–63.

[LYTM08] LAUTERBACH C., YOON S.-E., TANG M., MANOCHA D.: ReduceM: Interactive and memory efficient ray tracing of large models. *Computer Graphics Forum 27*, 4 (2008), 1313–1321.

[MB90] MACDONALD D. J., BOOTH K. S.: Heuristics for ray tracing using space subdivision. *The Visual Computer 6*, 3 (May 1990), 153–166.

[Mor66] MORTON G. M.: *A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing*. Tech. Rep. Ottawa, Ontario, Canada, IBM Ltd., 1966.

[Mor11] MORA B.: Naive ray-tracing: A divide-and-conquer approach. *ACM Transactions on Graphics 30* (October 2011), 117:1–117:12.

[MT97] MÖLLER T., TRUMBORE B.: Fast, minimum storage ray-triangle intersection. *Journal of Graphics Tools 2*, 1 (1997), 21–28.

[NFLM07] NAVRÁTIL P. A., FUSSELL D. S., LIN C., MARK W. R.: Dynamic ray scheduling to improve ray coherence and bandwidth utilization. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing* (Washington, DC, USA, 2007), RT '07, IEEE Computer Society, pp. 95–104.

[NIDN13] NABATA K., IWASAKI K., DOBASHI Y., NISHITA T.: Efficient divide-and-conquer ray tracing using ray sampling. In *Proceedings of the 5th High-Performance Graphics Conference* (New York, NY, USA, 2013), HPG '13, ACM, pp. 129–135.

[Nvi12a] NVIDIA: *CUDA C Programming Guide*, October 2012.

[Nvi12b] NVIDIA: *NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110*. Whitepaper, NVIDIA Corporation, 2012.

[ORM08] OVERBECK R., RAMAMOORTHI R., MARK W. R.: Large ray packets for real-time whitted ray tracing. In *IEEE/Eurographics Symposium on Interactive Ray Tracing 2008* (2008), pp. 41–48.

[PFHA10] PANTALEONI J., FASCIONE L., HILL M., AILA T.: PantaRay: fast ray-traced occlusion caching of massive scenes. In *SIGGRAPH '10: ACM SIGGRAPH 2010 papers* (New York, NY, USA, 2010), ACM, pp. 37:1–37:10.

[PGSS07] POPOV S., GÜNTHER J., SEIDEL H.-P., SLUSALLEK P.: Stackless kd-tree traversal for high performance GPU ray tracing. *Computer Graphics Forum 26*, 3 (2007), 415–424.

[PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory To Implementation*, 2nd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010.

[PKGH97] PHARR M., KOLB C., GERSHBEIN R., HANRAHAN P.: Rendering complex scenes with memory-coherent ray tracing. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 101–108.

[Rah13] RAHMAN R.: *Intel Xeon Phi Coprocessor Architecture and Tools: The Guide for Application Developers*. Apress Media LLC, New York, NY, USA, 2013.

[RL00] RUSINKIEWICZ S., LEVOY M.: QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), pp. 343–352.

[RL01] RUSINKIEWICZ S., LEVOY M.: Streaming QSplat: A viewer for networked visualization of large, dense models. In *I3D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2001), ACM, pp. 63–68.

[RSH05] RESHETOV A., SOUPIKOV A., HURLEY J.: Multi-level ray tracing algorithm. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 1176–1185.

[RW80] RUBIN S. M., WHITTED T.: A 3-dimensional representation for fast rendering of complex scenes. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1980), SIGGRAPH '80, ACM, pp. 110–116.

[SCS*08] SEILER L., CARMEAN D., SPRANGLE E., FORSYTH T., ABRASH M., DUBEY P., JUNKINS S., LAKE A., SUGERMAN J., CAVIN R., ESPASA R., GROCHOWSKI E., JUAN T., HANRAHAN P.: Larrabee: a many-core x86 architecture for visual computing. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 18:1–18:15.

[SE10] SEGOVIA B., ERNST M.: Memory efficient ray tracing with hierarchical mesh quantization. In *Graphics Interface 2010* (2010), pp. 153–160.

[SFD09] STICH M., FRIEDRICH H., DIETRICH A.: Spatial splits in bounding volume hierarchies. In *Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), HPG '09, ACM, pp. 7–13.

[SGNS07] SLOAN P.-P., GOVINDARAJU N. K., NOWROUZEZAHRAI D., SNYDER J.: Image-based proxy accumulation for real-time soft global illumination. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2007), PG '07, IEEE Computer Society, pp. 97–105.

[SHBS02] SZIRMAY-KALOS L., HAVRAN V., BENEDEK B., SZÉCSI L.: On the efficiency of ray-shooting acceleration schemes. In *Proceedings of Spring Conference on Computer Graphics (SCCG)* (2002), pp. 97–106.

[SM98] SZIRMAY-KALOS L., MÁRTON G.: Worst-case versus average-case complexity of ray-shooting. *Journal of Computing 61*, 2 (1998), 103–131.

[SM03] SHIRLEY P., MORLEY R. K.: *Realistic Ray Tracing*, 2 ed. A. K. Peters, Ltd., Natick, MA, USA, 2003.

[Smi98] SMITS B.: Efficiency issues for ray tracing. *Journal of Graphics Tools 3*, 2 (Feb. 1998), 1–14.

[SSK07] SHEVTSOV M., SOUPIKOV A., KAPUSTIN A.: Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes. *Computer Graphics Forum 26* (2007), 395–404.

[SSS08] SZIRMAY-KALOS L., SZÉCSI L., SBERT M.: *GPU-Based Techniques for Global Illumination Effects*. Morgan and Claypool Publishers, San Rafael, USA, 2008.

[SW13] SOMERS B., WOOD Z. J.: FlexRender: A distributed rendering architecture for ray tracing huge scenes on commodity hardware. In *GRAPP & IVAPP 2013* (2013), Coquillart S., Andújar C., Laramee R. S., Kerren A., Braz J., (Eds.), SciTePress, pp. 152–164.

[Szi08] SZIRMAY-KALOS L.: *Monte-Carlo Methods in Global Illumination — Photo-realistic Rendering with Randomization*. VDM, Verlag Dr. Müller, Saarbrücken, 2008.

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision* (Washington, DC, USA, 1998), IEEE Computer Society, pp. 839–846.

[TMG09] TORRES R., MARTÍN P. J., GAVILANES A.: Ray casting using a roped BVH with CUDA. In *Proceedings of the 25th Spring Conference on Computer Graphics* (New York, NY, USA, 2009), SCCG '09, ACM, pp. 95–102.

[Tsa09] TSAKOK J. A.: Faster incoherent rays: Multi-BVH ray stream tracing. In *Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), HPG '09, ACM, pp. 151–158.

[VBHH13] VINKLER M., BITTNER J., HAVRAN V., HAPALA M.: Massively parallel hierarchical scene processing with applications in rendering. *Computer Graphics Forum* (2013). To appear.

[Vea97] VEACH E.: *Robust Monte Carlo methods for light transport simulation*. PhD thesis, Stanford University, Stanford, CA, USA, 1997.

[VG97] VEACH E., GUIBAS L. J.: Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 65–76.

[Wal04] WALD I.: *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University, 2004.

[Wal07] WALD I.: On fast construction of SAH based bounding volume hierarchies. In *Proceedings of the 2007 Eurographics/IEEE Symposium on Interactive Ray Tracing* (2007), pp. 33–40.

[WBB08] WALD I., BENTHIN C., BOULOS S.: Getting rid of packets – efficient SIMD single-ray traversal using multi-branching BVHs. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing 2008* (2008), pp. 49–57.

[WBS07] WALD I., BOULOS S., SHIRLEY P.: Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Transactions on Graphics 26*, 1 (2007).

[WDS04] WALD I., DIETRICH A., SLUSALLEK P.: An interactive out-of-core rendering framework for visualizing massively complex models. In *EGSR04: 15th Eurographics Symposium on Rendering* (Norrköping, Sweden, 2004), Eurographics Association, pp. 81–92.

[WGBK07] WALD I., GRIBBLE C. P., BOULOS S., KENSLER A.: *SIMD Ray Stream Tracing - SIMD Ray Traversal with Generalized Ray Packets and On-the-fly Re-Ordering*. Tech. Rep. UUSCI-2007-012, SCI Institute, University of Utah, 2007.

[WH06] WALD I., HAVRAN V.: On building fast kd-trees for ray tracing, and on doing that in O(N log N). In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing* (2006), pp. 61–69.

[WMG*09] WALD I., MARK W. R., GÜNTHER J., BOULOS S., IZE T., HUNT W., PARKER S. G., SHIRLEY P.: State of the art in ray tracing animated scenes. *Computer Graphics Forum 28*, 6 (2009), 1691–1722.

[WRG07] WAGNER G. N., RAPOSO A., GATTASS M.: An anti-aliasing technique for voxel-based massive model visualization strategies. In *Proceedings of the 3rd International Conference on Advances in Visual Computing - Volume Part I* (Berlin, Heidelberg, 2007), ISVC'07, Springer-Verlag, pp. 288–297.

[WSBW01] WALD I., SLUSALLEK P., BENTHIN C., WAGNER M.: Interactive rendering with coherent ray tracing. *Computer Graphics Forum (Proceedings of EUROGRAPHICS) 20*, 3 (2001), 153–164.

[YGKM08] YOON S.-E., GOBBETTI E., KASIK D., MANOCHA D.: *Real-Time Massive Model Rendering*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, 2008.

[YLM06] YOON S.-E., LAUTERBACH C., MANOCHA D.: R-LODs: Fast LOD-based ray tracing of massive models. *The Visual Computer: International Journal of Computer Graphics 22*, 9 (2006), 772–784.

[YLPM05] YOON S.-E., LINDSTROM P., PASCUCCI V., MANOCHA D.: Cache-oblivious mesh layouts. *ACM Transactions on Graphics 24* (July 2005), 886–893.

[YM06] YOON S.-E., MANOCHA D.: Cache-efficient layouts of bounding volume hierarchies. *Computer Graphics Forum 25*, 3 (2006), 507–516.

[YSGM04] YOON S.-E., SALOMON B., GAYLE R., MANOCHA D.: Quick-VDR: Interactive view-dependent rendering of massive models. *IEEE Visualization* (2004), 131–138.

[ZL77] ZIV J., LEMPEL A.: A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory 23* (1977), 337–343.