Scalable many light methods

Martin Kahoun

8. dubna 2011

Úvod

V předchozí přednášce o *many light methods* jsme si popsali jejich princip a věnovali se detailně odstraňování jejich nedostatků. Nyní se podíváme, jak lze urychlit výpočet výsledků těchto metod – výpočet totiž vypadá zjednodušeně takto:

1. Vygeneruje se velké množství světel
2. Všechna světla se použijí k výpočtu osvětlení scény

Z toho nám vyplývá, že pro daný počet pixelů krát počet vygenerovaných světel musíme řešit viditelnost, což není triviální operace; dá se sice zrychlit použitím stínových map místo vrhání paprsků, ale pro velmi velký počet světel to není zase tak znát. Aplikací Wardovy metody [4] se můžeme zbavit vyhodnocování vržených stínů pro některá velmi slabá světla, nicméně i tak nám zůstává víceméně lineární závislost délky výpočtu na počtu světel, což může být problém, pokud jdeme řádově do tisíců až desetitísíců světel ve scéně.

Hledáním sublineární závislosti délky výpočtu na počtu světel se budeme věnovat nyní. Těmto metodám se říká škálovatelné a jejich princip je v podstatě následující:

1. Vygeneruje se *opravdu* velké množství světel
2. Vyberou se ta nejvhodnější světla pro výpočet osvětlení

Druhý bod nám samozřejmě opět do výskedků zanáší systematickou chybu (*bias*), což nám, jak si za chvíli ukážeme, zase tolik vadit nebude. Přístupy, jak se tohoto výběru zhostit jsou v zásadě dva: můžeme výběr provádět tzv. *per-pixel* nebo *per-image*. Ukážeme si oba přístupy.

Lightcuts

První metoda nazvaná *lightcuts*, neboli světelné řezy, spadá do kategorie *per-pixel* řešení výběru vhodných světel. Je podrobně popsána v [2]. Cena za nárůst počtu světel je sublieneární (viz graf v prezentaci) a umožňuje renderování scény osvícené například mapou prostředí (*environment map*), ze které po navzorkování vznikne skutečně hodně světel. Autoři tvrdí, že tato metoda umožňuje vcelku rychlé[1] renderování scén s počty světel v řádech milionů.

Základní myšlenkou metody je postavit nad světly binární strom – tzv. *light-tree*. Listy tohoto stromu odpovídají původně vygenerovaným světlům. Vnitřní vrcholy pak představují celý shluk světel (*light cluster*) – vybere se vhodné světlo ze shluku, jež bude sloužit jako jeho reprezentant. Intenzita tohoto reprezentanta je pak rovna součtu intenzit světel obsažených v tomto shluku (tedy intenzita levého a pravého podstromu, případně listů). Když máme strom postavený, tak na něm provedeme řez – tedy vybereme takové shluky světel či samotná světla, které vhodně aproximují osvětlení scény pro daný paprsek vržený do scény.

[1] Rychlým je míněn rozumný čas kratší nežli celá věčnost.

Stavba stromu

Než přistoupíme k samotnému výpočtu osvětlení, provedeme předzpracování scény: pomocí vhodného algoritmu (buď stejně jako jsme si ukazovali posledně, či vzorkováním mapy prostředí[2]) vygenerujeme virtuální světla. Nad vzniklým seznamem světel posléze postavíme binární strom. Lze jej stavět jak shora, tak zdola.

Jednodušší je přístup shora: seznamy dělím na dvě části podle nejdelší strany obalového objemu, případně dle mediánu. Je též vhodné shlukovat světla dle směru jejich normál; jen je nutné zohlednit, že normála není porovnatelná se souřadnicemi bodu ve 3D prostoru. Autoři navrhují přenásobit normály osminásobkem velikosti obalového objemu.

Výběr řezu

Pro každý paprsek vržený do scény od kamery musíme nyní vybrat vhodný řez stromem. Zde musíme nalézt rovnováhu mezi přesností příspěvku světla z řezu a "cenou" tohoto řezu – tedy jak hluboko a z kolika světel budeme příspěvek skládat. Zde nám pomůže Weberův zákon, který říká, že nejmenší pozorovatelná změna je přibližně fixní procento z celkového signálu. Autoři článku stanovili tento práh na 2% celkové luminance. To znamená, že maximální odchylka od referenčního řešení (výpočtem přes všechna virtuální světla) je maximálně 2%. Takováto chyba nebude pro běžného pozorovatele viditelná. Rovněž nebudou viditelné světelné artefakty mezi sousedními pixely, pro které algoritmus vybral jiný řez světelným stromem.

Algoritmus tedy pro všechny pixely obrázku prochází světelný strom a pro každý jeho vrchol (shluk světel) se snaží zaručit, že chyba osvětlení vůči referenčnímu obrázku, která vznikne nahrazením původních virtuálních světel tímto shlukem, je menší než 2%. Následující rovnice představuje výpočet referenčního obrazu O:

$$O = \sum_{i \in |\mathrm{L}|} M_i \cdot G_i \cdot V_i \cdot I_i \tag{1}$$

kde iterujeme přes všechna světla L, člen M_i odpovídá BRDF daného povrchu, G_i je geometrický člen, V_i představuje viditelnost na i-té světlo a I_i je intenzita i-tého světla. Pro aproximaci pomocí algoritmu *lightcuts* se nám rovnice (1) změní následovně:

$$O \approx M_j \cdot G_j \cdot V_j \cdot \sum_{i \in |\mathrm{L}|} I_i \tag{2}$$

index j zde značí reprezentativní světlo pro daný shluk (vrchol stromu).

Chceme-li vyjádřit maximální chybu aproximace rovnice (1) rovnicí (2), potřebujeme stanovit horní meze odchylek jednotlivých členů rovnice (2). Člen V_j se dá omezit hodnotou 1, protože víme, že bude vždy maximálně roven jedné. Intenzity jednotlivých světel jsou známy dopředu, zbývá tedy vyjádřit maximální odchylku pro zbylé dva členy. Autoři článku odvodili vztah závislý na velikosti obalového objemu. Maximální chyba aproximace E se dá tedy vyjádřit:

$$E = \bar{M}_j \cdot \bar{G}_j \cdot \bar{V}_j \cdot \sum_{i \in |\mathrm{L}|} I_i \tag{3}$$

kde $\bar{M}_j, \bar{G}_j, \bar{V}_j$ značí maximální odchylku členů rovnice (2) pro daný shluk. Snadno nahlédneme, že maximální chyba vnitřního vrcholu je stejná jako jeho maximální příspěvek k osvětlení scény.

Zbývá nám ovšem vyřešit jeden problém – neznáme referenční obraz, vůči kterému bychom mohli tuto chybu počítat (kdybychom ho znali, tak algoritmus ale pozbývá smysluplnosti). Můžeme ovšem provést následující trik: neznáme referenční obraz, nicméně v každém kroku počítáme jeho odhad, který se navíc zpřesňuje. Chybu příspěvků jednotlivých uzlů tedy můžeme porovnávat s tímto průběžným odhadem.

[2]Ideálně tak, že mapu prostředí prohlásíme hustotou pravděpodobnosti a dle toho vygenerujeme směrová světla z nichž následně vygenerujeme virtuální světla ve scéně.

Konečně tedy přistoupíme k výběru vhodných řezů stromem. Budeme si průběžně počítat odhad luminance s použitím vrcholů obsažených v řezu a začneme od kořene stromu. Zcela určitě bude chyba způsobená použitím jen tohoto vrcholu příliš velká a proto budeme pokračovat dále. Každý uzel, jehož chyba je větší než stanovená část průběžného odhadu (v našem případě 2%), vyřadíme z řezu a nahradíme jej jeho syny. Poté z řezu vybereme vrchol s nejvyšší chybovou metrikou a pokračujeme dokud v řezu existuje nějaký vrchol, jehož odchylka je větší než daný práh nebo dosáhneme maximálního počtu světel.

Zanesená chyba není pouhým okem vidět, a autoři uvádějí, že při zhruba desetinásobném zvýšení počtu světel se čas potřebný k vyrenderování obrázku jen zdvojnásobil. Největší slabinou tohoto algoritmu je přílišná velikost řezu na zastíněných místech.

Multidimensional lightcuts

Pokud bychom chtěli počítat například komplexní osvětlení, *anti-aliasing*, rozmazání pohybem, simulovat hloubku ostrosti či jiné efekty je nutné integrovat přes více domén. Každá taková doména vyžaduje více vzorků na jeden pixel výsledného obrazu a tím lineárně narůstá délka výpočtu (pokud bychom každý vzorek spojovaly s nějakým řezem skrze světelný strom). Autoři původního článku přišli o rok později s rozšířením, které je sublineárně škálovatelné i pro výpočty výše zmíněných efektů. Podrobnosti jsou uvedeny v [3]. Metodu nazvali *multidimensional lightcuts*.

Stejně jako v základní variantě *lightcuts* se vygenerují světla a postaví se nad nimi strom (*light-tree*). Následně je spuštěna výpočetní smyčka, která je však o něco složitější. Pro každý pixel se negeneruje jeden vzorek, ale celá řada vzorků, které autoři nazývají sběrné body (*gathering points*). Nad těmi se posléze postaví obdobný strom jako nad světly, který autoři nazývají sběrným stromem (*gathering-tree*). Pak je *de-facto* proveden kartézský součin obou stromů, nad kterým se teprve hledá řez. Řešení aproximuje následující rovnici:

$$O = \sum_{i,j \in \mathrm{G} \times \mathrm{L}} S_j \cdot M_{ji} \cdot G_{ji} \cdot V_{ji} \cdot I_i \tag{4}$$

Obraz tedy O vzniká jako součet přes všechny páry světel a sběrných bodů (L, G) přes součin členů známých z rovnice (1), kde přibyl ještě člen S_j odpovídající váze j-tého vzorku.

Implementačně se samozřejmě neprovádí kartézský součin stromů, místo toho se oba stromy prochází současně. Stav výpočtu je tedy určen pozicí v obou stromech a v každém kroku se vybírá člen s největší chybovou metrikou v jednom nebo druhém stromě. Méně triviální záležitostí jsou však chybové meze pro každý uzel stromů L, G. Pro horní omezení geometrického členu $G_j i$ se počítá minkowského součet obalových objemů j-tého shluku sběrných bodů a i-tého shluku světel. BRDF jsou vzorkovány do malých krychliček, na které jsou potom promítany obalové objemy světel. Více detailů o těchto postupech je v článku, jak zmiňuje prezentace.

Matrix row-column sampling

Druhá metoda škálování nazvaná *matrix row-column sampling* přistupuje k problému škálování *many lights* metod tzv. *per-image*. Autoři v [1] předkládají čistě navržený algoritmus, který umožňuje využít stínových map pro výpočty viditelnosti a na obraz hledí jako na celek.

Základní myšlenkou je sestavení matice příspěvků světla na jednotlivé body obrazu. Sloupcový prostor matice je tvořen všemi světly a řádkový prostor představuje všechny pixely cílového obrazu. Jeden řádek tedy představuje příspěvky všech světel k danému pixelu. Sloupec pak představuje příspěvky konkrétního světla ke všem pixelům obrazu. Výsledný obraz je pak výsledkem součtu všech sloupcových vektorů,

Podíváme-li se na sloupcové vektory, zjistíme, že prostor jimi tvořený je blízký prostoru s nízkou hodností, protože vektory jsou na sobě skoro lineárně závislé. Autoři tohoto faktu využívají při konstrukci váženého součtu vybraných a zredukovaných sloupců zmíněné matice, čímž aproximují výsledný obraz.

Vzorkování matice

Matici příspěvků světla chceme ideálně vzorkovat buď po řádcích či sloupcích, aby šlo využít stínových map k dalšímu urychlení výpočtu. Nejprve se náhodně vybere určité množství pixelů a vytvoříme z nich tzv. redukovanou matici, která bude sice dlouhá, ale nebude mít takovou výšku. O redukovaných sloupcích můžeme přemýšlet jako o obrázcích s velmi malým a hrubým rozlišením.

Redukované sloupce rozdělíme do shluků tak, že podobné sloupce stojí vedle sebe. Pro každý redukovaný sloupec se navíc zavede tzv. informační vektor, jež vznikne znormalizováním redukovaného sloupce a představuje jakýsi druh světelného příspěvku, jak to autoři nazývají. Dále se zavede váha sloupcového vektoru, což je jeho norma.

Naším cílem je provést shlukování těchto redukovaných sloupců a následný výběr vhodného vektoru, který bude daný shluk reprezentovat – reprezentant je vybírán s pravděpodobnostní úměrnou své normě. Výsledný obraz je poté aproximován váženým součtem neredukovaných reprezentantů jednotlivých shluků, což není nic jiného než Monte-Carlo estimátor.

Aby ale fungoval dobře, potřebujeme zajistit že bude mít nízkou varianci, musíme tedy utvořit "dobré" shluky. Pro názornost si nyní představme sloupcové vektory jako kroužky v nějakém mnoho-dimenzionálním prostoru, kde jejich poloměr odpovídá váze daných vektorů a pozice kroužků odpovídá pozicím informačních vektorů v tomto prostoru. Jako "dobré" shluky budeme uvažovat takové, které nebudou moc velké – řečeno matematicky, každému shluku přiřadíme jeho cenu a budeme se snažit minimalizovat celkovou cenu:

$$cena = \sum_{p=1\ldots k} \sum_{i,j \in C_p} w_i \cdot w_j \cdot \|\mathbf{x}_i - \mathbf{x}_j\|^2 \tag{5}$$

Jak vidno, sčítáme ceny všech možných shluků C_p, kde cena jednoho takového shluku je součtem přes všechny jeho dvojice vektorů; w odpovídá váze daného sloupcového vektoru a $\mathbf{x}$ odpovídá jeho informačnímu vektoru.

Minimalizace rovnice (5) je však NP-těžký problém, autoři se proto snaží jen o sub-optimální řešení pomocí kombinace čistě náhodného shlukování s metodou rozděl a panuj k dosažení použitelných výsledků (jen náhodné vzorkování má tendenci vytvářet příliš velké shluky).

Výsledná chyba obrazu opět není pro běžného pozorovatele viditelná, avšak renderovací čas se posouvá z řádů minut do řádů vteřin.

Reference

[1] Hašan M. et al, *Matrix row-column sampling for the many-light problem.* Proc. SIGGRAPH '07.

[2] Walter B. et al, *Lightcuts: a scalable approach to illumination.* Proc. SIGGRAPH '05.

[3] Walter B. et al, *Multidimensional lightcuts.* Proc. SIGGRAPH '05.

[4] Ward G., *Adaptive Shadow Testing for Ray Tracing*, Eurographics Workshop on Rendering, 1991.