Global Illumination – výpisky

Pre-computed radiance transfer II

Aleš Zita

Úvod

V minulé kapitole jsme si ukázali, jakým způsobem lze aproximovat funkci na kouli řídkou reprezentací s užitím sférických harmonických funkcí. Pomocí této reprezentace jsme dokázali redukovat úlohu výpočtu přímého osvětlení na povrchu materiálu na pouhý jeden skalární součin s tím, že jsme přenos radiance na povrch dokázali reprezentovat jediným číslem (v případě ideálně difuzního povrchu), nebo několika koeficienty sférických harmonických bázových funkcí (pro povrch s obecnou BRDF).

V této kapitole se budeme zabývat metodami syntézy obrazu, které mohou běžet v reálném čase, nebo alespoň budí dojem, že v reálném čase běží. Zaměříme se na metody pracující s komplexním osvětlením scény, stíny a případně také globálním osvětlením scény.

Vzhledem k tomu, že čas potřebný pro rendering typické scény na dnes běžně dostupném hardwaru je stále za hranicí realtimové obrazové syntézy, jediný způsob, jakým lze dosáhnout snímkovací frekvence, jež by se dala považovat za realtime, je najit způsob, jak rozdělit náročný výpočet na dvě fáze. V první fázi, kterou budeme nazývat *offline* fází, budeme provádět časově náročné operace, jakýsi předvýpočet tak, aby bylo možné počet operací druhé fáze (vlastní *rendering*) zredukovat na počet, který dokáže běžet v reálném čase.

Metoda realtime renderingu

V předchozí kapitole jsme při obrazové syntéze uvažovali pouze přímé osvětlení povrchu. Nyní bychom rádi přidali také stíny nebo nepřímé osvětlení do kalkulace, ale s tím, že zachováme real-time povahu obrazové syntézy. Ve fázi předvýpočtu budeme chtít spočítat hodnoty, které zafixujeme (typicky geometrii, BRDF), abychom v druhé fázi dokázali opět redukovat výpočet do nějaké relativně rychlé operace. V této budeme schopni měnit scénu jen velmi omezeně, typicky její osvětlení.

V našem případě převedeme druhou fázi výpočtu, tedy vlastní obrazovou syntézu. Vzhledem k tomu, že zobrazovací rovnice je lineární vůči radianci, závisí barva daného pixelu na osvětlení lineárně. Důsledkem je, že lze rendering scény vyjádřit například jako násobení *matice s vektorem*. Výsledkem této operace pak bude například vektor pixelů tvořící výsledný obrázek.

Př. : Lze například sestavit zmíněnou matici tak, že pokud touto vynásobím zleva vektor reprezentující osvětlení ve formě mapy prostředí, výsledkem této operace bude rendering scény. Takovou operaci díky

tomu, že jsou matice i vektor osvětlení typicky rozměrné, nelze provádět opakovaně s frekvencí, jež by se dala nazvat interaktivní.

Nadále se v této lekci budeme zabývat tím, jak takovouto matici vytvořit, případně jak jí zjednodušit.

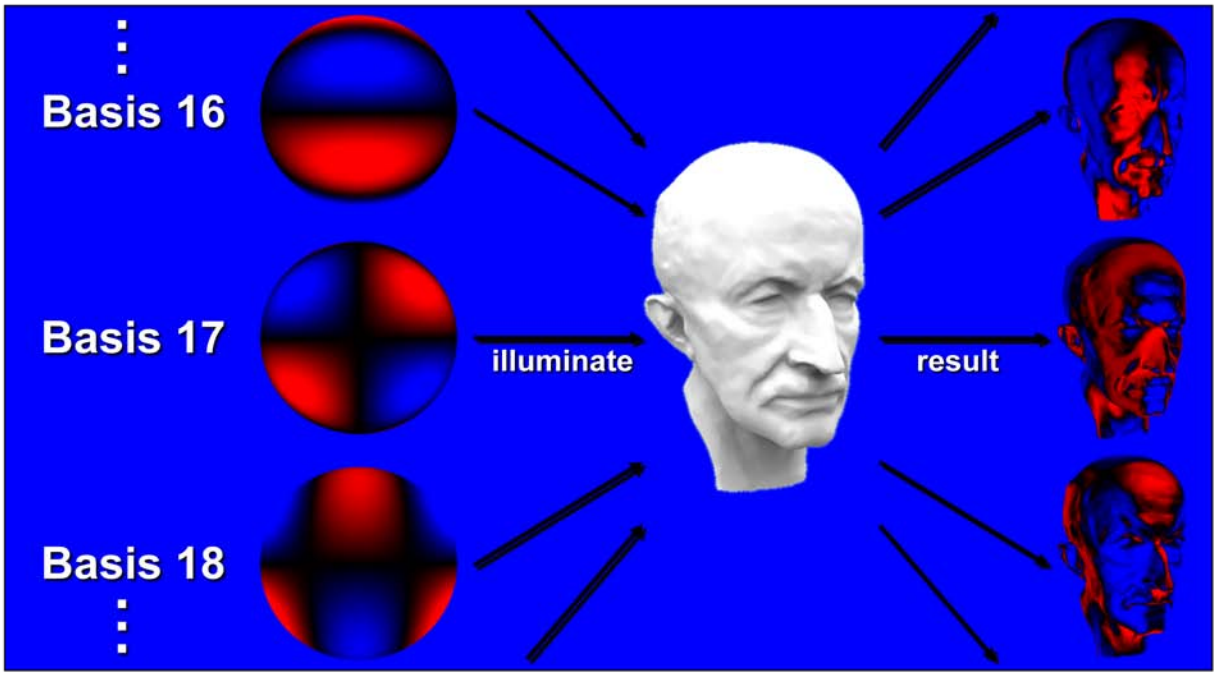
Protože je takto definovaná matice enormně veliká, tudíž počet operací prováděných v každém snímku se může pohybovat v řádech 10^{10}, představíme způsoby redukce její velikosti při zachování co největší podobnosti. Redukcí velikosti matice vyřešíme pochopitelně nejen prostorovou náročnost takto definované operace, ale také dosáhneme snížení počtu operací.

Kompresní metody

Řídká reprezentace pomocí SH

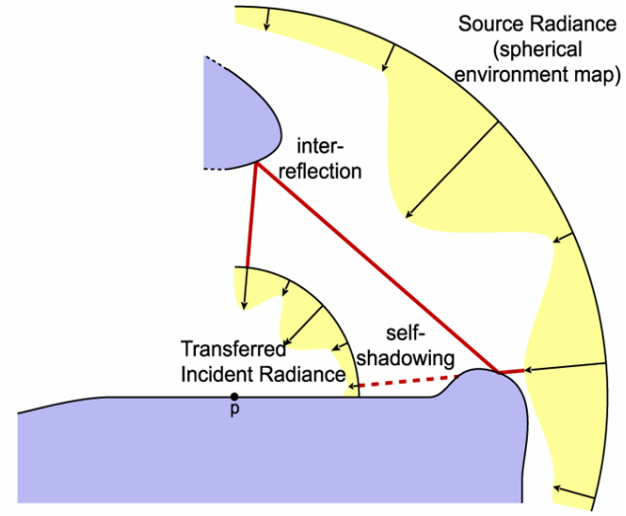
Do této kategorie patří například řídká reprezentace pomocí koeficientů *Sférických Harmonických Funkcí* (dále jen SH) na podobném principu, jako v minulé kapitole. Vektor osvětlení zde bude reprezentován koeficienty Sférických harmonických funkcí, a tudíž bude matice reprezentující scénu tvořit bázi SH. V důsledku komprese dané povahou SH, tak jak jsme si ukázali v minulé kapitole lze vektor osvětlení redukovat na několik málo koeficientu (obecně platí, že 25 je dostatečná aproximace, pokud jsou povrchy scény blízké difuzním).

Sloupce matice bází budou takto reprezentovat obrázek osvětlený světlem aproximovaným jedním koeficientem a řádky jednotlivé body scény. Výsledný obrázek je pak součtem zobrazení pomocí jednotlivých koeficientu SH. Viz Obrázek 1.



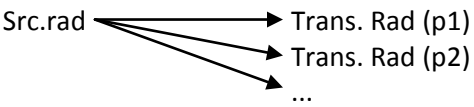
Obrázek 1. Reprezentace osvětlení scény pomocí koeficientů sférických harmonických funkcí.

Vektor reprezentující osvětlení vlastně reprezentuje funkci radiance, která do scény vstupuje bez stínů a nepřímého osvětlení. Naopak výsledný vektor představuje radianci v konkrétním bodě scény. Viz Obrázek 2.



Obrázek 2. Definice pojmů - Source Radiance = radiance vstupující do scény například ve formě mapy prostředí, Transferred Incident Radiance = radiance v určitém bodě scény.

Celý rendering se pak vlastně dá vyjádřit jako funkce, která má jako vstupní parametr vektor radiance do scény vstupující a výsledkem jsou radiance v jednotlivých bodech scény :



Výše popsanou úvahou lze upravit matici tak, že bude mít pouze 25 řádků (při použití 25 koeficientů), což je velikost vektoru osvětlení. Počet řádků pak určíme jako 25*počet bodů u kterých se osvětlení počítá. Pro případ difuzních povrchů nám stačí pro reprezentaci Transferred Radiance pouze jediné číslo, neboť již není potřeba údaje o příchozím směru radiance, a netřeba ji tedy reprezentovat jako funkci na sféře. V takovém případě bude mít matice počet řádků roven počtu bodů scény, pro kterou předvýpočet provádím.

Samotný rendering difuzního povrchu tedy vypadá tak, že maticovým násobením dostaneme Transfererd Radiance pro určité vrcholy scény. Výslednou barvu pixleu pak získáme tím, že interpolujeme z příslušných hodnot Transferred Radianci na příslušném místě obrázku a vynásobíme výslednou hodnotu odrazivostí daného materiálu, tedy texturou. Pro případ povrchu lesklého mi PRT pomocí maticového násobení dá pro každý bod oněch 25 koeficienů Transferred Radiance. Poté můžeme pro vyhodnocení barvy pixelu použít například metodu Kautz 2003 (popsaná v minulé přednášce), kde pomocí skalárního součinu koeficientů BRDF a Transferred Radiance získám výslednou hodnotu odražené radiance.

PCA/SVD

Dalšími technikami komprese matice, které bychom zde mohli zmínit, jsou metody založeny na principu SVD faktorizace (Vzorec 1), která dává stejné výsledky jako metoda Principal Component Analysis (viz Vzorec 3).

$$A = U\Sigma V^T$$

Vzorec 1. SVD faktorizace. A - faktorizovaná matice; U,V - matice ortonormální; Σ - matice diagonální.

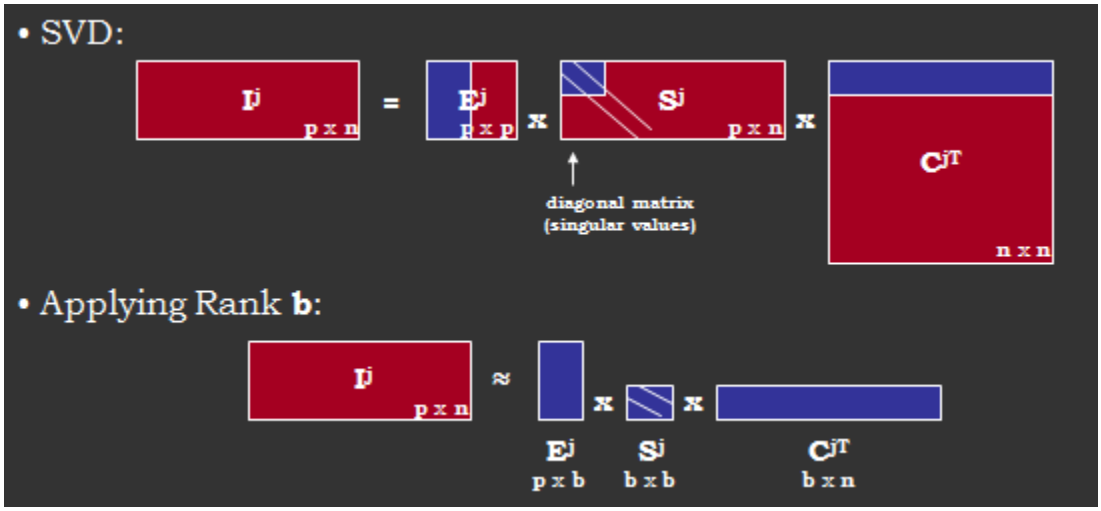
$$AA^T = \Lambda \mathrm{L} \Lambda^T$$

Vzorec 2. PCA. Λ - je matice ortonormální; L - matice diagonální.

$$AA^T = (U\Sigma V^T)(U\Sigma V^T)^T = U\Sigma V^T V\Sigma^{\mathrm{T}} U^T = U\Sigma\Sigma^{\mathrm{T}} U^T = \Lambda\mathrm{L}\Lambda^T$$

Vzorec 3. Náznak důkazu, že PCA ≡ SVD.

SVD faktorizace vychází z faktu, že lze každou matici rozepsat jako součin dvou ortogonálních matic a jedné diagonální. U diagonální matice využijeme toho, že v důsledku faktorizace jsou její singulární hodnoty seřazeny dle velikosti. Postupným vynecháváním singulárních hodnot od nejmenší po největší nejen postupně snižujeme dimenzi diagonální matice, ale navíc tím také získáváme dobré aproximace původní matice (po zpětném roznásobení všech třech složek). Lze dokázat, že jsou tyto aproximace v L2 normě nejbližší možné. Tímto způsobem lze reprezentovat původní matici maticemi s námi zvolenou dimenzionalitou (přesností). Tímto se součin třech velkých matic změní na součin jedné malé diagonální matice a dvou „hubených“ matic (viz obrázek.), který reprezentuje nejbližší možnou aproximaci původní matice.



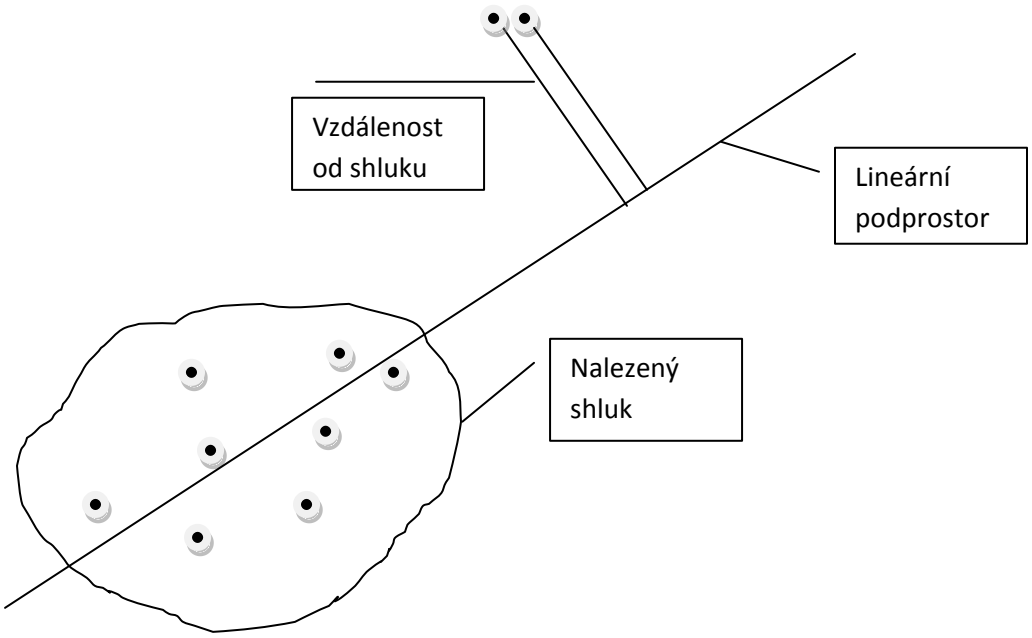
Obrázek 3. SVD komprese.

Bohužel přímá aplikace této metody nepřináší dobré výsledky. Byť je metoda teoreticky v pořádku, v případu syntézy obrazu není matice přenosu dobře podobná žádné matici nižší hodnosti. Je tedy potřeba matici pod-rozdělit na menší části, u kterých již nebude problém podobnosti dosáhnout. Toho docílíme použitím stejného principu s malou obměnou.

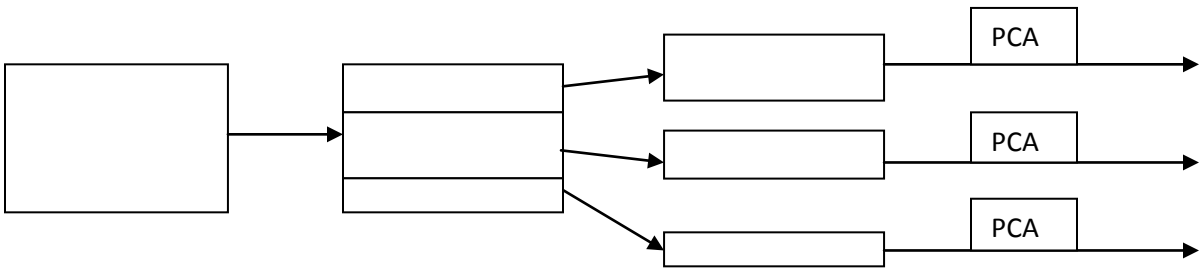
CPCA

Tato metoda vychází z principů dvou komprimačních algoritmů: Vektorové kvantizace a PCA. Půjde v ní o rozdělení řádků matice (t.j. pixelů, ev. vrcholů meshe) do „shluků“ na které poté budeme

aplikovat metodu PCA komprese individuálně. Účelem bude podrozdělení na situace, v rámci kterých je již možné PCA kompresi použít aniž bychom se dopustili velké chyby. Tato metoda se nazývá *CPCA(Clustered PCA)*. Shluky nalezneme upravenou metodou shlukové analýzy nazývané *k-means*. K-means pracuje na principu iterativního vyhledávání těžiště shluku a následnou úpravou hranic tak dlouho, dokud středy shluků nezkonvergují. V našem případě však nebudou hranice shluků určeny euklidovskou vzdáleností tak, jak je tomu v klasické verzi tohoto algoritmu, ale pro naše účely definujeme vzdálenost bodu od shluku jiným způsobem. Hlavní myšlenka tkví v tom, že jako zástupce shluku nebudeme brát jeho těžiště, ale jeho aproximaci lineárním podprostorem výrazně nižší dimenze. Dimenze tohoto podprostoru bude stejná, jako počet nenulových singulárních hodnot, které zachovám po SVD (typicky zadáno uživatelem, např. 8). Tento podprostor je tudíž právě podprostor daný výsledkem SVD dekomozice. Vzdálenost bodu od shluku bude tak definována jako euklidovská vzdálenost od jeho ortogonální projekce do podprostoru. Viz Obrázek 4.



Obrázek 4. Upravený K-Means algoritmus, který používá definici vzdálenosti jako vzdálenost od lineárního podprostoru reprezentujícího shluk.



Obrázek 5. Princip CPCA - Původní matice je rozložena na bloky "podobných" řádků a PCA faktorizace je prováděna na každém bloku zvlášť.

Vytvořené shluky reprezentují nějaký blok matice (shluk řádků). Pokud na každý takovýto blok použijeme metodu SVD komprese, získáme visuálně mnohem lepší aproximaci než pokud bychom použili SVD na celou matici.

Pozn. Existují také metody, např. Hierarchické matice (H-matrices), které nepracují s celými řádky, ale místo toho matici hierarchicky dělí a poté aproximují takto rozdělené části.