Participating media (2. časť)

Martin Kahoun, Mária Vámošová

9. května 2011

Nehomogénne médium

V nehomogénnom médiu nemôžme analyticky spočítať inverziu distribučnej funkcie transmitancie, preto musíme použiť iné metódy generovania vzdialenosti paprskov podľa dôležitosti. Môžme na to ísť nasledujúcimi spôsobmi:

Vygenerujeme si náhodné číslo ξ, a hľadáme, v akej vzdialenosti $T = \xi$. Toto môžme urobiť pomocou ray-marchingu - v krokoch počítame hodnotu transmitancie, až kým neprekročíme hodnotu ξ, a potom interpolujeme v danom intervale medzi krokmi. V každom kroku si ešte skontrolujeme, či sme náhodou nenarazili na plochu.

Pseudokód:

Zavedieme si pomocnú funkciu na nájdenie vzdialenosti, kde $T = \xi$.

h je veľkosť kroku pri ray-marchingu, s_0 je počiatočná vzdialenosť a T_0 je transmitancia v počiatočnej vzdialenosti. T_1 je hodnota transmitancie, ktorej vzdialenosť hľadáme.

function find_attenuation(s_0, T_0, T_1, h)
    $\tau = -ln(T_0)$
    $\tau_1 = -ln(T_1)$
    for $(s = s_0; \tau < \tau_1; s+ = h)$
        $f = \sigma_t(s)$
        $\tau+ = hf$
    $s- = (\tau - \tau_1)/f$
    return s

Celý algoritmus potom vyzerá nasledovne:

h je veľkosť kroku, v ktorom nasčítávame utlmenie, a *trace_ray* vráti vzdialenosť a radianciu najbližšieho priesečníka. ϵ je emisný koeficient v danom bode. rand() je funkcia na vygenerovanie náhodného čísla uniformne z intervalu $(0, 1)$.

function radiance_estimator(x, ω):
    $s_{max}, L_0 =$ trace_ray()
    $\xi =$ rand()
    $s =$ find_attenuation$(0, 0, \xi, h)$
    if $(s < s_{max})$ then
        return $\epsilon(x - s\omega)/\sigma_t(x - s\omega)$
    else
        return L_0

Druhou možnosťou je tzv. Woodcock tracking - obdoba ruskej rulety. Na začiatku si spočítame σ_{max} ako maximálne σ_t cez všetky miesta v médiu. Pri generovaní vzdialenosti sa potom k médiu správame, akoby bolo homogénne s transportným koeficientom σ_{max} - takže vygenerujeme vzdialenosť analyticky ako:

$$s = \frac{-ln(1 - \text{rand}())}{\sigma_{max}}$$

Túto vzdialenosť potom prijmeme s pravdepodobnosťou:

$$\frac{\sigma_t(x - s\omega)}{\sigma_{max}}$$

alebo pokračujeme v generovaní vzdialenosti od tohoto bodu. Čitateľ zlomku je vlastne transmitancia v bode kam sme sa presunuli.

Pseudokód:

function sample_distance (x, ω, σ_{max}):
    $s = 0$
    while **true**:
        $s+ = -ln(1 - \text{rand}())/\sigma_{max}$
        if (rand() $< (\sigma_t(x - s\omega)/\sigma_{max})$) then
            break
    return s

Bidirectional Path-tracing

Na začiatok si sformulujeme rovnicu transportu svetla ako integrál cez všetky možné cesty v scéne:

$$I_j = \int_{\omega} g_j(\overline{x}) d\mu(\overline{x})$$

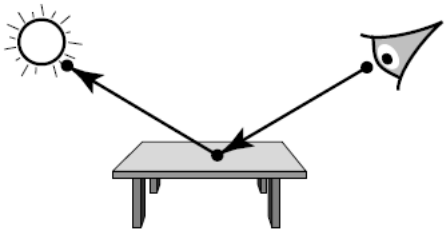
kde ω je priestor všetkých ciest medzi svetelnými zdrojmi a pixelom j, $\overline{x}$ je jedna konkrétna cesta určená svojimi bodmi, $g_j(\overline{x})$ je príspevok cesty $\overline{x}$ k pixelu j, a $\mu(\overline{x})$ značí mieru na množine svetelných ciest.

Pre spočítanie svetelného integrálu pomocou metódy Monte-Carlo potrebujeme vygenerovať svetelné cesty pomocou nejakej metódy, spočítať ich príspevok, a vedieť hustotu pravdepodobnosti, s akou tieto cesty generujeme. Potom náš odhad svetelného integrálu je:

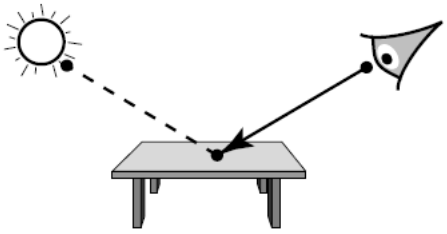
$$I = \frac{1}{N} \sum_{i=1}^{N} \frac{g(\overline{x_i})}{p(\overline{x_i})}$$

kde $g(\overline{x_i})$ je príspevok cesty i, a $p(\overline{x_i})$ je hodnota hustoty pravdepodobnosti, s akou bola vygenerovaná táto cesta.

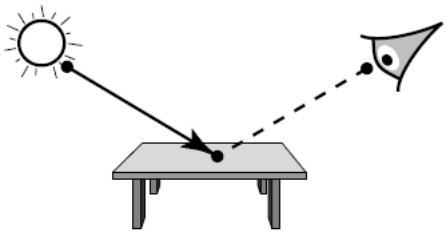
Bidirectional path-tracing generuje svetelné cesty kombináciou path-tracingu a sledovania ciest od svetla. Vygenerujeme cestu od kamery, ako v prípade path-tracingu (popis vyššie), a taktiež vygenerujeme cestu od svetla. Konečná cesta potom vznikne spojením koncov cesty od kamery a od svetla. Túto cestu si ale môžeme predstaviť ako viacero ciest, ktoré vznikli rôznymi kombináciami koncov, ale vo výsledku sú rovnaké.



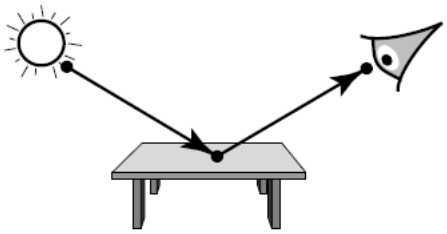
(a) $s = 0, t = 3$



(b) $s = 1, t = 2$



(c) $s = 2, t = 1$



(d) $s = 3, t = 0$

Pre odhad integrálu kombinuje BPT Monte-Carlo estimátory pre rôzne spôsoby generovania ciest - rôzny spôsob odpovedá rôznej dĺžke podtrasy cesty od kamery, a podtrasy cesty od svetla. Tieto cesty majú síce každá rovnaký príspevok (keďže sú to vo výsledku rovnaká cesta), ale každý takýto spôsob generovania má inú hustotu pravdepodobnosti. Tieto cesty skombinujeme pomocou metódy *Multiple importance sampling*.

Pre BPT na plochách platí, že príspevok cesty je rovný

$$g(\overline{x}) = \prod_i [f(x_i) \cdot G(x_i, x_{i+1})] \cdot L_e \cdot W_e$$

kde x_i sú vrcholy cesty, f je BRDF vo vrcholoch, G sú geometrické členy úsekov medzi vrcholmi, L_e je emitancia svetelného zdroja, a W_e je odozva kamery.

Hustota pravdepodobnosti cesty sa rovná súčinu hustôt s akými sme generovali jednotlivé vrcholy cesty.

V objeme zostáva princíp algoritmu nezmenený. Cesty od kamery a od svetla ale generujeme pomocou metód popísaných vyššie. Tým sa nám mení príspevok cesty, aj jej pravdepodobnosť následovne:

$$g(\overline{x}) = \prod_i [f(x_i) \cdot G(x_i, x_{i+1})] \cdot L_e \cdot W_e$$

rovnako ako pri BPT na plochách, kde ale

$$f(x_i) = \begin{cases} f_r(x_{i-1} \to x_i \to x_{i+1}) & \text{ak je vrchol na ploche} \\ \sigma_s(x_i) \cdot p(x_{i-1} \to x_i \to x_{i+1}) & \text{ak je vrchol v médiu} \end{cases}$$

a

$$G(x, y) = \frac{D_x \cdot D_y}{||x - y||^2} \cdot \tau(x, y) \cdot V(x, y)$$

kde

$$D_x = \begin{cases} \cos\theta_x & \text{ak je vrchol na ploche} \\ 1 & \text{ak je vrchol v médiu} \end{cases}$$

Čo sa týka hustoty pravdepodobnosti na priestore vygenerovaných ciest, tú spočítame rovnako ako u BPT na plochách - ako súčin hustôt pre jednotlivé vrcholy, kde hustota pre vrchol je:

$$p(x_i) = p(\omega) \cdot p(s)$$

kde ω je vygenerovaný smer, a s vygenerovaná vzdialenosť v tomto smere.

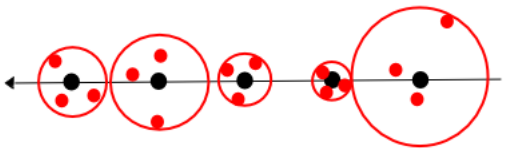
Volumetrické fotonové mapy

Posledným zmieneným spôsobom sú volumetrické fotónové mapy, ktoré sú analógiou plošných fotónových máp. Myšlienka rozšírenia je taká, že budeme generovať náhodné prechádzky pomocou metód popísaných vyššie, a ukladať fotóny pri každej rozptylovej udalosti. Pri každom fotóne si potom zapamätáme informáciu o smere, odkiaľ prišiel, a o svetelnom toku, ktorý nesie. Potom môžeme fotóny buď priamo použiť ako VPL, alebo implementovať klasický photon mapping.

V prípade photon mappingu je ale následné zbieranie energie komplikovanejšie, pretože musíme akumulovať výsledky z fotónov umiestnených pozdĺž cesty trasovaného paprsku.

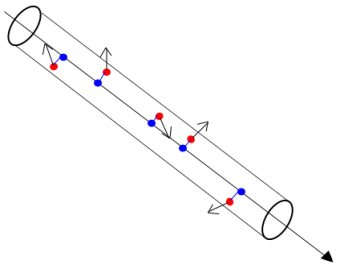
Toho se dá docieliť buď pomocou už zmieneného *ray-marchingu*, kde musíme pre každý vzorek urobiť klasický k-NN (k-nearest neighbor) dotaz do fotónovej mapy. Nerobí sa ale už final-gathering, pretože v obecnom heterogénnom médiu je veľmi drahý, a v médiu tolerujeme väčšie optické chyby ako na plochách.

Alternatívne se dá použiť tzv. *beam radiance estimate*, kde se zbierajú fotóny pozdĺž celého paprsku z kamery naraz (takže stačí jediný dotaz). Ten je síce efektívny, ale zato pomerne komplikovaný. Spočíva v tom, že sa vygeneruje valcovitý paprsok v danom smere, s určitým



Obrázek 1: k-NN dotaz do photon mapy

polomerom, a do úvahy sa berú fotóny, ktoré padnú do vnútra tohoto valca. Tie sa premietnu do priamky prechádzajúcej stredom valca, a spočíta sa fázová funkcia medzi smerom fotónu (odkiaľ prišiel) a naším smerom. Váhová funkcia váži príspevok každého fotónu jeho vzdialenosťou od stredu valca.



Obrázek 2: Beam marching

Nevýhodou beam radiance estimate oproti ray marchingu je, že nemôžeme marchovať adaptívne (v blízkej vzdialenosti malé kroky, v diaľke veľké kroky). Berie do úvahy všetky fotóny vo vnútri valca, aj keď sú ďaleko, a teda ich príspevok sa stratí. Riešením je kombinácia metód: beam radiance estimate do nejakej vzdialenosti, a potom riedky ray-marching.

Ďalšou variantou photon mappingu je interpretovať trasu fotónov ako valcovité paprsky, ktoré potom pretíname paprskom od kamery. Môžeme to urobiť progresívne: zmenšovať priemer valca - zo začiatku budeme mať rozmazaný obrázok s malým šumom, ktorý nám dá dobrý odhad, a postupne k nemu budeme pričítať výsledky renderu s menším polomerom, a priemerovať - týmto odhad spresňujeme.

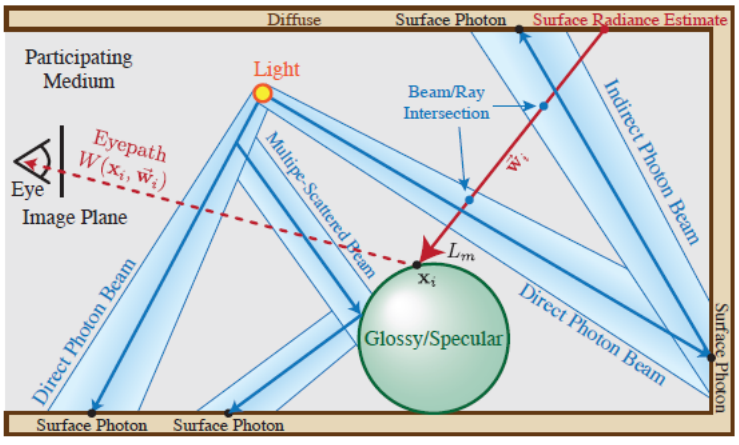
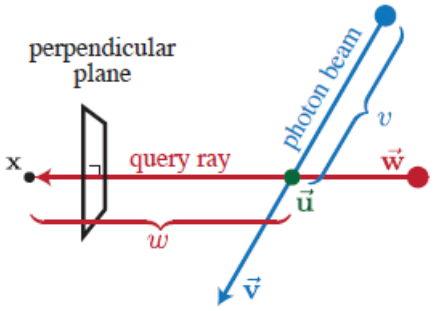


Figure 3: *We first shoot photon beams (blue) from the light sources (yellow). We then trace paths from the eye (red) until we hit a diffuse surface, estimating volumetric scattering by finding the beam/ray intersections and weighting by the contribution W to the camera. In each pass we reduce a global radius scaling factor, and repeat.*

Obrázek 3: Photon Beams



Obrázek 4: Photon beams schéma

Ak by sme stále mali počítať transmitanciu úseku v, bolo by to veľmi neefektívne. Tento problém sa dá ale riešiť pomocou techniky Deep Shadow Maps [DSM]. DSM je vlastne klasický Shadow Maps, kde máme ale namiesto jednej hodnoty uloženú nejakú reprezentáciu funkcie (napríklad koeficienty). Ak máme potom transmitanciu uloženú v DSM, tak sa pre úsek v rovno len pozreme do mapy, a nemusíme nič počítať. Artefakty tejto metódy sú paprsky - ktoré ale aj tak očakávame, takže vo finále to vyzerá pekne.

Zhrnutie: Metóda photon mappingu je *biased* a je treba ukladať ohromné množstvo fotónov, aj tak ale ide o robustné, jednoduché a rýchle rozšírenie algoritmu pre rendrovanie B-rep scény. Tento prístup je hojne používaný.