フォトンマッピング法による照度分布図生成と高速化

谷口　総一朗
Soichiro TANIGUCHI

1　はじめに

我々は，オフィスにおいて，執務者の希望する光環境を実現する知的照明システムの研究を行っている [1)]．現在，オフィスへのシステム導入実験 [2)] も行っている．そこで，導入前のシミュレーション，システムのデモンストレーションを行うという点から，実システム無しで知的照明システムの研究を可能とするような，シミュレータの必要性は高いといえる．また，知的照明システムのような照度値の分布を参照したいというシステムへのシミュレータの実現に関して，照度分布図表示が必要と考えられる．知的照明システムのような，複雑な照明環境下での明るさ算出を行う際には，様々な照明環境下で明るさ算出する手法を利用する必要がある．より正確な照度分布図生成のため，直射光による照度値だけではなく間接光による照度値も算出する必要がある．

本研究では照度分布図生成に，フォトンマッピング法を用いる．また，フォトンマッピング法は計算量が多く処理時間が長い一方で高い並列性を持っているため，高速化について検討・実装し，性能について評価を行う．

2　フォトンマッピング法

2.1　計算手順

フォトンマッピング法は，大きく分けて二つのアルゴリズムから構成される．

第 1 段階として，光源からフォトンをランダム方向に放射する．フォトンとは，光のエネルギーと，空間中の位置データを持つ仮想の光子である．放射されたフォトンは，様々な物体と交差し，物体の材質によって反射・屈折・吸収の状態変化を起こす．この時，フォトンが交差した空間中の物体表面の座標，フォトンの光束を保存したデータをフォトンマップと呼ぶ．

第 2 段階として，フォトンマップを用いて，照度値を算出したいある点について，照度推定を行う．このある点をクエリと呼ぶ．照度推定のイメージを Fig. 1 に示す．クエリの照度推定には，クエリの近傍フォトンの光束を用いて，光束密度を推定することで行う．以下の式 (1) で照度値 E の推定を行う．

$$E = \frac{\sum(\phi_i)}{\pi R^2} \tag{1}$$

照度値は単位面積あたりの光束量で算出される．そのため，光束量 ϕ_i をもったフォトンを用いてクエリ付近の合計光束量と，クエリから最も離れている近傍フォトンとの距離 R を用いて面積を算出し，式 (1) のように近似できる．

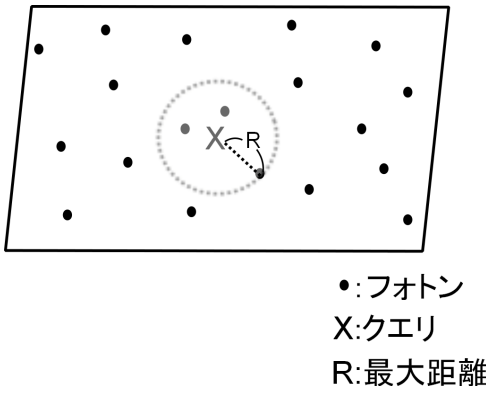

Fig.1　照度推定のイメージ

2.2　フォトンマップのデータ構造

フォトンマップから，クエリ近傍のフォトンを探索する際に，全てのフォトンを参照して近傍かどうか判定を行う方法を全探索法と呼ぶ．しかし，全探索法による近傍探索は，クエリの近傍になりえないフォトンも参照している．そのため, クエリの近傍フォトンだけを参照するような, 空間分割データ構造でフォトンマップを管理する必要がある．一般的に，フォトンマップを管理する空間分割データ構造は KD 木が用いられる [3)]．KD 木の構築手順を以下に示す．また，空間イメージを Fig. 2，構築イメージを Fig. 3 に示す．なお，フォトンマップの空間次元は 3 次元だが，簡単のため，2 次元上のデータを用いるものとする．

Fig. 2 は，黒い点がフォトンで，それぞれに番号をふった空間分割イメージである．Fig. 3 は Fig. 2 で用いたフォトンに対し，順番に KD 木を構築している．

1. フォトン集合について, ある軸と垂直な分割面により, フォトンを分割する．分割面により，ある軸を 2 分する．また，この分割面はフォトンを通るように選択する．一般的に，バランス木になるように木を構築するため，フォトン群のある軸成分の中央値にあたるフォトンを通るように分割面を決定する．
2. 木の構築を行う．分割面と接するフォトンを分割ノードとする．分割面の軸成分よりもフォトンの軸成分が小さいフォトン集合を左の子ノードに, 大きいフォトン集合を右の子ノードとする．
3. 子ノードに含まれるフォトン数が一つになるまで,1,2 を繰り返し，子ノードを分割する．この際, ある軸を x → y → z と変化させ, 分割面を選択する．

照度推定については，KD 木で管理しているフォトン

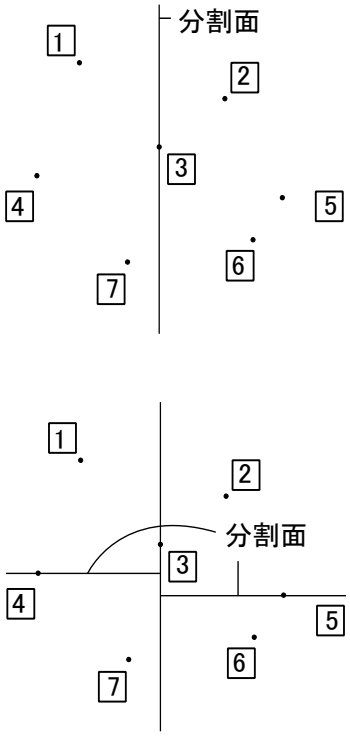

Fig.2　空間分割イメージ

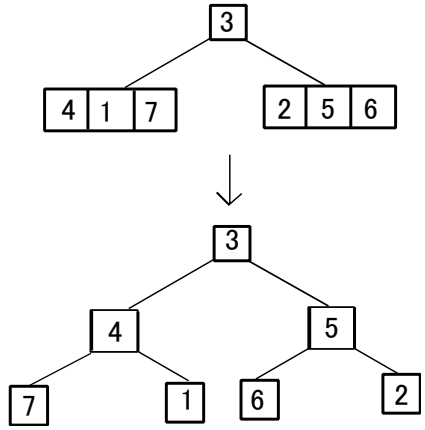

Fig.3　KD 木構築イメージ

マップから，クエリからの探索距離 d を指定し，近傍フォトンを探索する．探索手順を以下に示す．なお, 推定フォトンリストの要素数は推定フォトン数である．Fig. 4 に，KD 木のフォトン探索イメージ図を示す．

Fig. 4 は，KD 木をどのようにたどり，フォトンリストに対し，どのようにフォトンを管理し，近傍フォトンを採択するかを示した図である．この図では，フォトン推定に 3 つのフォトンを用いるものとした．

1. KD 木のルートノードから探索する．
2. ノードに格納されているフォトンとクエリから, 距離を算出し, 推定フォトンリストに追加する．
3. 探索距離を用いて, 探索距離内に入る空間の子ノードへ探索を行う．両方の子ノードを探索する必要がある場合, 左側の子ノードを優先的に探索し, もう一方の子ノードは, 左の子ノードの葉ノードまでの探索を終えると, 探索が行われる．
4. 2,3 を繰り返し行う．推定フォトンリストが埋まっている際に，新たなフォトンを推定フォトンリストに追加するには，以下の処理を行う．
   (a) クエリと新たに追加するフォトンとの距離 rr を算出する．
   (b) 推定フォトンリスト内のフォトンの最大距離 r を算出する．
   (c) r と rr の距離を比較する．
   (d) rr が r よりも小さければ r のフォトンと入れ替える．この際, 探索距離 d に新たな最大距離を算出し設定する．
5. 探索される予定の子ノードについて, 葉ノードまで探索が行われたとき, 探索終了となる．

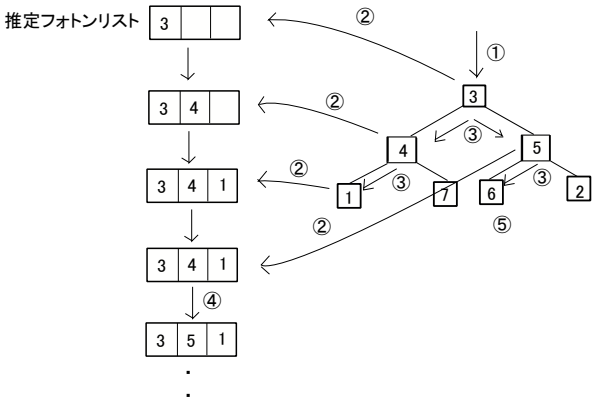

Fig.4　KD 木探索のイメージ

2.3　並列性

フォトンマッピング法は，フォトン探索部分の計算量が多く処理時間が長い一方で高い並列性を持っていると報告されている [4]．そのため，並列処理による高速化が考えられる．

3　実装

フォトン探索部分は，画素ごとに処理が独立している．このことから，フォトン探索処理を画素並列に処理する実装を行う．並列化にあたり，OpenMP を用いた実装を行った．

4　評価

本研究で提案した手法の評価を行う．フォトンマッピング法は変化させるパラメータが複数あるため，それぞれのパラメータを変化させた時の高速化率を評価する必要がある．画素数 120 × 144 ，推定フォトン数 100，放射フォトン数 3 万個を基準とし，それぞれのパラメータを 2 倍ずつ増加させ，フォトン探索にかかる時間を計測した．そして，シングルスレッド実行の C と 4threads 実行の OpenMP との処理時間を比較し，高速化率を算出した．実行時のマシン環境を，Table1 に示す．

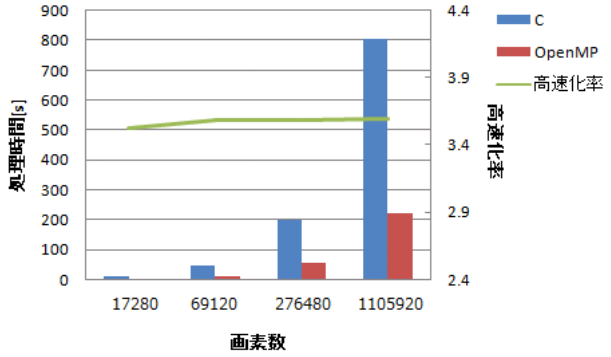

Fig.5　画素数と処理時間の関係図

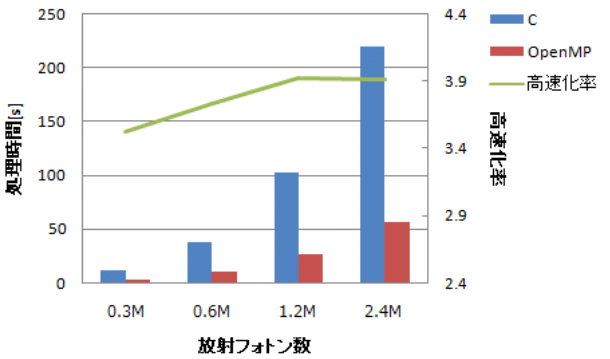

Fig.6　放射フォトン数と処理時間の関係図

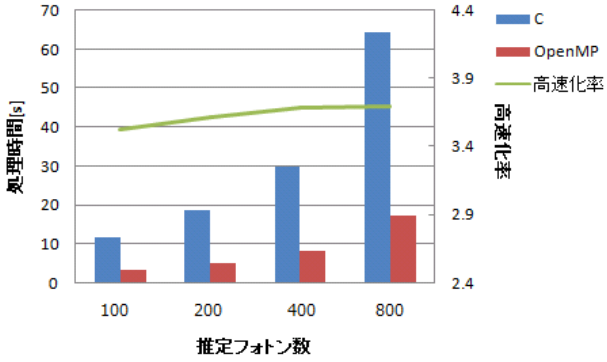

Fig.7　推定フォトン数と処理時間の関係図

画素数を変化させたときの C と OpenMP との処理時間の関係を調査し，比較した図を Fig. 5 に示す．同様に，放射フォトン数，推定フォトン数を変化させた時の処理時間の関係を調査し，比較した図を Fig. 6，Fig. 7 に示す．

単純に 4 倍のスレッド数を用いていることから，4 倍の高速化率が考えられる．結果を見ると，それぞれの図から，約 3.5〜4 倍の高速化が確認できた．Fig. 5，Fig. 6，Fig. 7 の高速化率を比較すると，高速化率が 4 倍に近づくときは，放射フォトン数を増加させたときといえる．また，それ以外の場合の高速化率は約 3.5 倍となっていることがわかる．このような現象が起きる原因としては，ループ処理を実行するスレッドを生成するためのオーバーヘッドが全体の処理時間に対して支配的な大きさになることによるものと考えられる．放射フォトン数が増加するほど，画素ごとの計算部分の実行時間が増大していく．放射フォトン数が大きい場合では，計算部分の実行時間が支配的な大きさになり，高速化率が大きくなったものと考えられる．

5　今後

4threads による並列実装を行ったので，今後はさらに複数のスレッドによる画素並列実装が考えられる．そこで，より多くの thread で並列処理が可能な画像処理専用ハードウェア GPU(Graphics Processing Unit) を用いた GPGPU(General-Purpose computing on GPU) での画素並列処理実装が考えられる．

Table1　評価に使用したマシン実行環境

OS	Ubuntu 12.10 x86_64
Memory	8 GB
CPU	Intel Core i5-2400 3.10GHz 4threads
CPU code Compiler	g++
gcc version	4.6.3
compile option	-O3

6　まとめ

フォトンマッピング法による照度分布図生成を行った．また，フォトンマッピング法は，フォトン探索部分の計算量が多く処理時間が長い一方で高い並列性を持っているため，並列化による高速化が考えられる．フォトン探索処理は，画素ごとに処理が独立しているため，画素並列実装を行った．そして，シングルスレッド実行の C と，4threads 実行の OpenMP による処理時間をそれぞれ計測し，高速化率を算出し評価を行った．複数スレッドを用いて高速化できたことを確認できたため，今後は，より多くの thread を用いて並列処理可能な GPU を用いて，画素並列処理実装が考えられる．

参考文献

1) 三木光範. 知的照明システムと知的オフィス環境コンソーシアム，人工知能学会誌，vol.22，no.3，pp.399-410，2007.
2) M. Miki, F. Kaku, T. Hiroyasu, M. Yoshimi, S. Tanaka, J. Tanisawa, , T. Nishimoto. Construction of intelligent lighting system providing desired illuminance distributions in actual office environment, journal of the institute of electronics, information and communication engineers of japan, vol.j94-d, pp.637-645，2011.
3) H.W.Jensen. Realistic image synthesis using photon mapping,a.k.peters,ltd.,natick,ma,usa,2009.
4) 久原拓也, 吉見真聡, 三木光範. fpga を用いたフォトンマッピング法高速化手法の提案と性能評価.