GPU を用いた 大規模点群の平滑化

東京大学　○池田邦彦，増田宏

Smoothing of a Large-Scale Point-Cloud with GPU

The University of Tokyo: Kunihiko Ikeda, Hiroshi Masuda

The recent progress of mid/ long-range laser scanners enables to capture large-scale point-clouds in short time. However, since such a point-cloud contains many large noises and outliers, reliable smoothing methods are very important. Although the moving least-squares and the moving robust estimate are powerful smoothing tools, they are very time-consuming for large-scale point-clouds. In this paper, we propose a GPU-based method to accelerate the moving robust estimate. We implemented it on a GPU and evaluated its performance. The result shows that our GPU-based method is significantly fast compared to CPU-based smoothing.

1. はじめに

中・長距離計測可能なレーザスキャナが急速に進歩し，大規模点群を容易に得られるようになってきた．そうした点群を用いて，現実に則した 3D モデルを作成できれば非常に有用である．

しかし，中・長距離のレーザスキャナで計測されたデータは，図 1 に示すように多くのノイズや異常値を含む．そのため，モデリングには平滑化が非常に重要となる．しかし，平滑化で用いられている移動最小２乗法や移動ロバスト推定[1]では，各点の平滑化に非線形最適化が必要となるため，非常に多くの計算時間がかかる．

近年，計算の高速化のために GPU がよく利用されており， GPU を汎用的な計算に用いた GPGPU も注目されている．そこで本研究では，大規模点群の平滑化に GPU を用いることで，実行時間を短縮する手法を検討することにする．

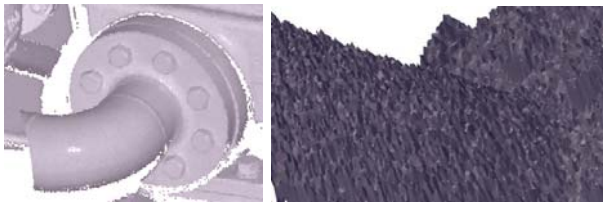


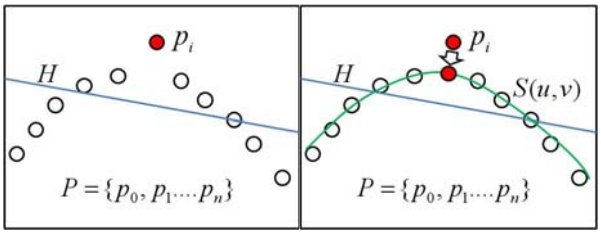
Fig. 1　Captured point data (left: point-cloud, right: zoom-up of mesh)

2. 点群の平滑化

本研究では，ロバスト推定に基づく平滑化手法[1]を考える．この方法では，まず，各点の近傍の点群に当てはまる参照平面 (図 2(a)) を計算する．各点は，参照平面上に投影され，ハイトフィールド (u,v,z) に変換される．その後， $z=S(u,v)$ となる u,v の 2 次曲面 $S(u,v)$ を算出する (図 2(b))．この関数は以下の式を解くことで得られる．ここで， a は曲面の係数ベクトル， σ は標準偏差， p_c は平面上の基準点， θ は距離によって減衰する関数である．

$$\min_a \sum_i \left[\log\left\{1+\frac{S(u_i,v_i)^2}{\sigma^2}\right\}\theta\left(\left|p_c-q_i\right|\right)\right]$$

平滑化では，元の点を $S(u,v)$ 上に投影して座標を補正する．この計算をすべての点に適用することで，平滑化された点群が得られる．



(a) referenced plane: H　　(b) quadric surface: S

Fig. 2　Smoothing by fitting a quadratic surface

3. GPU による平滑化

本研究では NVIDIA の GPU を用い，実装には CUDA を用いる．

3.1 GPU のアーキテクチャ

GPU は多くのスレッドを瞬時に切り替えることで，データを大量に並列処理する能力に優れ，高い浮動小数点演算能力を持つ．CUDA プログラミングではそのスレッドを，スレッドの集合をブロック，ブロックの集合をグリッドとして管理する (図 3)．

一方，ハードウェアとしての GPU 内部には，8 個の SP (Streaming Processor)をもつ SM (Streaming Multi Processor) がいくつも存在

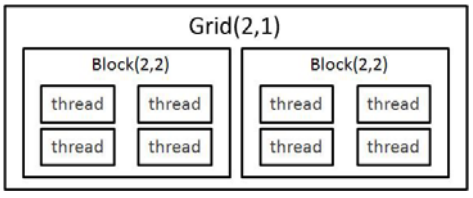


Fig. 3　Threads on GPU

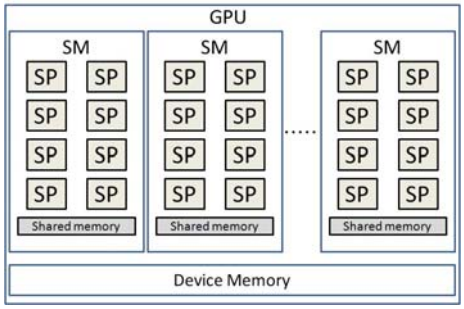


Fig. 4　GPU Architecture

する．SM にはレジスタ並みに高速なシェアードメモリが存在し，SP がこれを共有する．スレッドはそれぞれブロック単位で SM に割り当てられるので，同じブロック内のスレッドはシェアードメモリを共有できる．デバイスメモリはすべての SP からアクセスでき，容量も大きいが，レイテンシがレジスタやシェアードメモリに比べて非常に大きく，キャッシュもないので頻繁に使うデータはシェアードメモリかレジスタに置くことが望ましい．

3.2 GPU を用いた平滑化の方法

2 章で述べた平滑化を GPU で行うことを考える．この平滑化では各点の処理を独立に行え，データが規則正しく並んでいるので，画像処理と類似した並列化が可能である．また，それぞれの平滑化で必要となる近傍点に対してはほぼ同様な計算をするので，近傍点に関する計算も並列化することができる．

そこで本研究では，1 つの点の平滑化を 1 つのブロック(SM と言い換えてもよい)で行い，近傍点 1 つに対してスレッド 1 つを割り当てる．また，平滑化の計算では近傍点の座標に頻繁にアクセスするので座標値はシェアードメモリで保持しておく (図 5).

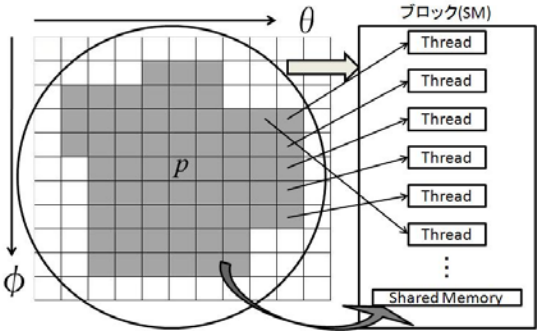


Fig. 5 Smoothing of point p on GPU

3.3 GPU での平滑化

GPU の平滑化の流れは以下に示す通りである．

(1) 点群データの転送

カーネルでの 1 回の平滑化処理に必要な点群データを CPU のメモリ(ホストメモリ)から GPU のメモリ(デバイスメモリ)に転送する．

(2) カーネル実行時のスレッド数とブロック数

3.2 で述べたようなスレッドの割り振りを行うので，カーネルの実行時のブロック数は平滑化を行う点の数だけ作成し，各スレッド数は参照する近傍点の数だけ作成する (図 5).

(3) シェアードメモリへのデータ転送

1 つのブロックは 1 点の平滑化を担当するので，平滑化に必要な近傍点をシェアードメモリにコピーする．

(4) 無効な点や距離の離れた点の削除

点群データでは，計測に失敗した点は座標が (0,0,0) となっているので，それらの点を削除する．また，近傍点の数が閾値を下回る場合も無効な点とする．さらに，注目点から遠すぎる点についても削除する．有効なデータは，その後の計算でスレッドにおける分岐が少なくなるように，シェアードメモリの先頭側に寄せておく．

(5) 近傍点から参照平面，二次曲面の算出

近傍点の準備が終わったら 2 章で示した平滑化を行う．注目点からの距離に応じた重み関数や座標変換などの計算は各点で独立に行えるので各スレッドが担当して行う．一方で，参照平面や二次曲面のパラメータ計算での初期解を求める際や，ガウスニュートン法で解を更新していく際，頻繁に，連立一次方程式 $AX = B$ を解く必要がある．この A, B の要素は，暫定解の値や近傍点の値を元に算出された値の総和などで計算される。すでに定まっている値(例えば暫定解の値や残差の値)から要素の値を計算する場合には，スレッド単位で並列化することはせず，スレッド1のみで要素の値を計算する．一方で，近傍点のデータを元に算出された値の総和で求められる要素に関しては，まず担当している各スレッドに値を計算させ，それらの総和を，総和計算用に確保したシェアードメモリ上でスレッド同士を同期させながら計算していく．同様に，近傍点の情報に対して総和計算が必要なもの(分散や残差など)の計算を行う際にはスレッド 1 を中心に協調させながら計算する．そうして求められた方程式 $AX = B$ をスレッド 1 で解く(図 6).

(6) 計算結果の CPU への転送

カーネル関数の実行が終わったら CPU へ結果を転送する．

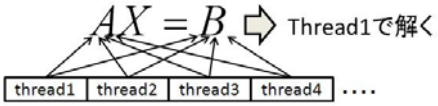


Fig. 6 Calculation of AX=B

3.4 実行結果と実行時間

601×301 の点群に対して，GPU を用いて平滑化をした結果(一部拡大)を図 7 に，本手法の実行時間と従来の実行時間(Core2 シングルコア)の比較を表 1 に示す．GPU を用いて平滑化が行うことができ，また，平滑化の速度を飛躍的に向上させることができている．

Fig. 7 Result of smoothing (left: original, right: smoothed)

Table. 1 Timing of smoothing operations

	CPU (Core2 2.66GHz)	GPU (GTX285)
実行時間(秒)	5.4×10 sec.	3.8 sec.

4. まとめ

本稿では，GPU を用いて，大規模点群の平滑化を高速化する方法を示した．また，実装して評価することで，その有効性を示した．

今後の課題として，各閾値の値などで平滑化の結果が変わってくるので，計算に用いる閾値の検証や，その閾値の自動設定法などを検討する必要である．また，点群の平滑化には二次曲面当てはめのみではなく，球面当てはめを用いたものもあるので，他の曲面当てはめへの応用も考えてみたい．なお，本研究は科学研究費補助金(21360069)の助成を受けて行われた．

参考文献

[1] 大規模点群データの平滑化手法に関する研究（第 1 報）－ ロバスト推定に基づく平滑化手法 －: 増田宏，村上健治，精密工学会論文誌, 76(5), 2010.

[2] NVIDIA CUDA Programming Guide: NVIDIA, 2009