GPGPU を活用したモーションデータの楽曲同期システムにおけるマッチング処理の高速化に関する一検討

八木良一†　　柳原広昌†

（株）KDDI 研究所†

1．はじめに

近年グラフィックス・ハードウェア（GPU）の分野では，コンピュータ・グラフィックス（CG）処理のためのパイプライン化や，複数の並列演算器，MIMD（Multi Insutruction/ Multiple Data），SIMD（Single Instruction/ Multiple Data）命令といったハイパフォーマンスコンピューティングのための高性能演算機能が組み込まれるなど，特に浮動小数点数の演算性能において，CPU の数十倍の演算能力を実現している．一方，GPU を CG 以外の汎用演算に利用する GPGPU（General Purpose GPU）[1]の研究[2]も盛んに進められ，携帯端末用 GPGPU を搭載したプラットフォームや，高級言語に対応した NVIDIA 社提供の独自並列プログラミング環境（CUDA（Compute Unified Device Architecture））が登場するなど，様々な用途，目的に応じた環境も整備されつつある．

そこで，本稿では，GPGPU を活用した処理高速化の一検討として，モーションデータの楽曲同期システムにおけるモーションデータの動き特徴量と，音楽データの楽曲特徴量のマッチング処理の高速化方式を提案し，その有効性を検証したので報告する．

2．モーションデータの楽曲同期システム

（1）概要

CG 研究分野では，動きデータとして自然なキャラクタアニメーションを自動生成するために，MOCAP（Motion Capture）システムによって得られた MOCAP データを利用する手法が提案されている[3]．その一つに音楽データの特徴量と楽曲の盛り上がり情報を解析し，その結果に適合した MOCAP データを抽出することで，情感豊かな一連のキャラクタアニメーションを自動生成する手法[4]がある．以下に従来手法の処理手順を示す．

① 動作・音楽解析及び特徴量の抽出
② 楽曲特徴量に基づく MOCAP データの抽出
③ 連結可能性の解析及び連結候補の選定
④ 盛り上がり情報によるヒストグラムマッチング
⑤ 連結動作の生成

（2）楽曲特徴量に基づく MOCAP データの抽出

従来手法の処理手順②は，式(1) [4] の演算結果(スカラー値)である相関値を評価することによって適合度を判定して行う．式(1)において，$F_R^{Music}(f;M)$は楽曲セグメント M の楽曲特徴量，$F_R^{Motion}(s \cdot f+f0)$は MOCAP データの動き特徴量，$L_{Motion}$は MOCAP データの長さ，$L_{Music}$は楽曲セグメント M の長さとする．また，MOCAP データの切り出し長さは，スケールパラメータ s により伸縮を可能とし，抽出区間の開始フレームを f_0とする．この式(1)を用いて楽曲セグメントの楽曲特徴量との相関値が最も高くなるスケールパラメータ s と，その時の相関値 $\hat{s}$を算出する．

$$\hat{s} = \arg\max_{s} \sum_{f=0}^{L_{music}} \frac{F_R^{Music}(f;M) \cdot F_R^{Motion}(s \cdot f + f_0)}{F_R^{Music}(f;M) + F_R^{Motion}(s \cdot f + f_0)} \quad (1)$$

$$f_0 \in [0,\ L_{motion} - L_{music}]$$

しかしながら，この式(1)は，全ての MOCAP データと，全ての楽曲セグメントに対し，それぞれ演算を実行する必要があり，膨大な演算回数を必要とする．また，楽曲セグメントと動きセグメントの一致性を向上する上でも，動きデータベースのデータ登録件数と，楽曲セグメント数は一定数以上を保持する必要がある．

例えば 300 秒の音楽データから 300 個の楽曲セグメントを抽出し，1 秒当りのフレーム数を 120，スケールパラメータの個数を 10 個とする．一件当たり 3 秒の長さを持つ 1,000 個の MOCAP データがある場合に，一つの音楽データに対し行われる抽出処理の演算回数は，300×1,000×10×(360−120)=720,000,000 回となる．これは 3.6G FLOPS 程度の処理に相当する．さらに，同期処理全体に占める抽出処理の割合を 50%と仮定し，これを 1 秒で処理したい場合には 7.2G FLOPS 程度の CPU が必要と試算できる．

（3）高速化対象とするマッチングアルゴリズム

一般的に，予め楽曲特徴量をパターン化し，それに合致する動きセグメントを参照することで演算の実行回数を削減するマッチング手法が考えられるが，楽曲セグメントと動きセグメントの一致性が低下する可能性がある．

そのため，本稿では，パターン化を行わないマッチングアルゴリズムの高速化実装手法を検討した．

3．提案手法

前述の問題点に対する実装上の一つの手法として，GPGPU と CUDA プログラミングモデルの特徴を考慮した高速化手法を提案する．

なお，本提案手法では，動きデータベースの MOCAP データと音楽データから抽出した楽曲セグメントを CUDA の共有メモリにセグメント単位で階層的にマッピングすることで抽出処理の並列性を高め，高速化を図る．

図 2 に，CUDA プログラミングモデルを用いた楽曲特徴量に基づく MOCAP データの抽出処理のシステム構成例を示し，以下に処理手順を概説する．

① 前処理として，MOCAP データを特徴量に変換し，動きデータベースに保存する．
② 動きデータベースから MOCAP データの特徴量を取得し，共有メモリに保存する．

"A Fast Method for Automatic Synthesis of Motion Data Using GPGPU"
† Ryoucihi Yagi, Hiromasa Yanagihara.
KDDI R&D Laboratories Inc.

③ 音楽データを特徴量に変換し，共有メモリに保存する．
④ CPU 側から相関値演算の実行を指示する．
⑤ 共有メモリから MOCAP データと音楽データの特徴量を取得し，相関値演算を並列実行する．
⑥ 共有メモリに保存された相関値を取得し，MOCAP データと音楽データの特徴量が最も高い相関値を得る．

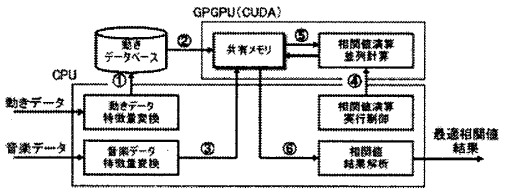

図 2　システム構成例

さらに，図 3 において，CUDA のプログラミングモデルを使った具体的なメモリアクセスの実装例を示し，以下に詳説する．

前処理として，音楽データを音楽解析し，長さ L_{Music} の楽曲セグメントに分割する．次に，MOCAP データベースより MOCAP データを取得し，L_{Music} にスケールパラメータの倍率を掛け，動きセグメントの切り出し長を決定する．さらに，MOCAP データの切り出し位置を 0 から i までとし，動きセグメントの切り出しを行う．以上の手順により生成された楽曲セグメント 1 個と動きセグメント N×10×i 個を共有メモリに格納する．共有メモリに格納された楽曲セグメントと動きセグメントとの相関値演算をスレッド上に配置し並列実行する．処理完了後，演算結果を共有メモリへ格納し，楽曲セグメントと最も相関の高い動きセグメントを抽出する．

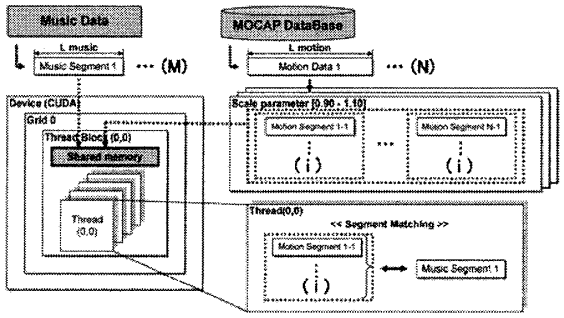

図 3　CUDA プログラミングモデルの実装例

- 音楽データの楽曲セグメント数を M 個とする．
- MOCAP データベースのデータ登録数を N 個とする．
- MOCAP データの長さを L_{Motion} とする．
- 楽曲セグメントの長さを L_{Music} とする．
- $L_{Motion}>L_{Music}$ が成立する場合，L_{Motion}-L_{Music}=i とする．
- スケールパラメータは 0.90 ～ 1.10 の範囲において，0.02 単位でインクリメントされるものとする．

4. 検証と考察

提案手法の有効性を確認するため，以下条件にて検証を行い，楽曲特徴量に基づく MOCAP データの抽出処理の処理時間を，CPU で実行した場合と提案手法で実行した場合とで比較した．なお，検証環境は CPU：Pentium4 3.8GHz，メモリ：2GB，GPU： NVIDIA 社製 GeForce GTX 280 GPU を搭載した ELSA 製 GLADIAC GTX 280 1GB ボードを使用した．

(1) 条件

44.1kHz，16bit，stereo，PCM，15 秒の音楽データを入力した．なお，音楽データの音楽解析により，5 個の音楽セグメントに分割されるとする．また，動きデータベースの MOCAP データは 15 個，GPGPU 上で実行する最大スレッド数は 512 個とし，楽曲セグメントと動きセグメントの基準フレームレートは 120fps とした．

検証結果を図 4 に示す．

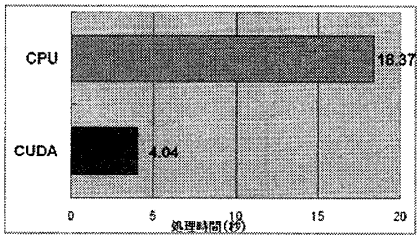

図 4 検証結果（相関値計算）

図 4 より，提案手法では CPU 単体で実行した場合と比べ，約 1/4 の処理時間で当該処理を実行できることが分かる．従って，処理全体に占める当該処理の比率を 50%と仮定した場合に約 38%の削減効果を期待できる．

5. おわりに

GPGPU を活用した処理高速化の一検討として，モーションデータの楽曲同期システムにおけるモーションデータの動き特徴量と，音楽データの楽曲特徴量のマッチング処理の高速化方式を提案し，その有効性の検証を行った．検証の結果，CUDA プログラミングモデルを利用することで，CPU 単体の実行時と比べ，処理時間を，約 1/4（処理全体としては約 62%）に削減できることが確認できた．また，CUDA プログラミングモデルを有効利用するためには，数百以上のスレッドを同時生成し，かつ高速な共有メモリを最大限に活かす従来の並列プログラミングとは異なる実装が必要であることも判明した．

今後は，MOCAP データの連結可能性の解析処理や盛り上がり情報のヒストグラムマッチング処理へと GPGPU の適用範囲を広げ，さらなる高速化を図る．

参考文献

[1] H. Nguyen 編：”GPU Gems 3”, Addison-Wesley, pp.765-907, 2007.

[2] 内之宮 仁志,西尾 孝治,小堀 研一：” GPU を用いたボリューム細分割の高速化に関する研究”,情報処理学会第 69 回全国大会講演論文集, 5Y-1, 第 4 分冊 , pp.245-246, 2007.

[3] L.Kovar,M.Gleichaar,and F.Pinhin：”Mottion graphs”,ACM Trans. Graphics (Proc.ACM SIG-GRAPH 2002),vol.21,no.3, pp.473-482,2002.

[4] 白鳥 貴亮,中澤 篤志，池内 克史：”音楽特徴を考慮した舞踊動作の自動生成”,電子情報通信学会論文誌,D,Vol.J90-D,No.8,pp.2242-2252,2007.