N,yVlN0[PX1O[fO
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

O[fbX1
IEICE Technical Report
CPSY2022-40,DC2022-99(2023-03)

Jetson Xavier NX における ORB-SLAM3 の低消費電力化の検討

林　　頼人[†]　見神　広紀[††]　納富　　昭[††]　木村　貞弘[†††]　木村　啓二[†]
笠原　博徳[†]

† 早稲田大学
†† オスカーテクノロジー株式会社
††† 株式会社エヌエスアイテクス
E-mail: †rhayashi@kasahara.cs.waseda.ac.jp

あらまし　環境地図の作成を行いながら自己位置の推定を行う SLAM は，自動車の自動運転やロボットの制御に必須の技術である．これらの機器が適切に制御されるためには，ある一定時間内に SLAM 処理を可能とするだけの計算能力が必要である．その一方で，これらの機器は設置スペースやバッテリ能力が制限されていることが多く，低消費電力化も重要な課題となっている．本稿では，SLAM 処理の代表的な実装の一つである ORB-SLAM3 に対して，その中の主要処理の一つであるトラッキングに着目し，本トラッキング処理に要する時間を処理中に得られた値から推定し，推定コストにより CPU の適切な動作周波数・動作電圧を設定する手法を提案する．そして，本手法を並列化した ORB-SLAM3 に実装し，Nvidia 製の組み込み SoC ボードである Jetson Xavier NX 上で評価した結果について報告する．制限時間を超過したフレーム数と消費電力を評価した結果，提案手法は Linux 標準の電力制御ガバナである Performance に比較して 71.8% の消費電力で同等の超過フレーム数を達成し，また Ondemand ガバナと比較して 86.6% の消費電力で 99.2% の超過フレーム数削減を達成できた．さらに，提案手法を加重移動平均（WMA）予測器による制御と比較して，提案手法は 90.5% の消費電力で同等の超過フレーム数を達成した．

キーワード　SLAM，Jetson，低消費電力化

Preliminary Evaluation of Low Power Optimized ORB-SLAM3 on Jetson Xavier NX

Raito HAYASHI[†], Hiroki MIKAMI[††], Akira NODOMI[††], Sadahiro KIMURA[†††], Keiji KIMURA[†], and Hironori KASAHARA[†]

† Waseda University
†† Oscar Technology Corporation
††† NSITEXE,Inc.
E-mail: †rhayashi@kasahara.cs.waseda.ac.jp

1. はじめに

自動車の自動運転やドローン，さらにはロボットが現実社会で広く利用されつつある．これらの自律制御する機器では，環境地図の作成を行いながら自己位置の推定を行う SLAM が必要不可欠となっている．特に，Visual SLAM は安価なカメラにより位置推定が可能なため広く用いられている．

しかしながら，SLAM には画像処理やグラフ処理などの計算負荷の高い処理が必要なため，これをリアルタイム処理するためには計算機システムに高い演算処理能力が要求される．その一方で，SLAM を必要とする機器はその自律移動するという性質上バッテリの容量に制限があり，また上記 SLAM 処理を行う計算機システムを設置するスペースも大きく確保できなかったり廃熱機構の設置が困難な場合もある．すなわち，必要とされる演算処理能力を低消費電力で実現するという相反する要求を満たさなければならない．

コンピュータシステムの消費電力削減のため，動的な周波数・電圧制御（DVFS）やクロックゲーティング，及びパワー

ゲーティングといった電力制御のためのハードウェアが広くコンピュータシステムで利用可能となっており，これらのハードウェア制御機構を前提とした様々な低消費電力手法が提案されている [1], [2]．SLAM のようなデッドライン制約を持つリアルタイム処理に対しても，適切な動作周波数と動作電圧を設定することで，システムの消費電力を抑えつつ必要な演算能力を確保する手法が提案されている [3]〜[6]．この時，適切な動作周波数を設定するためには処理中のタスクのコストを正確に推定することが重要である．

本稿では，代表的な SLAM の実装の一つである ORB-SLAM3 [7] に対して，その主要処理の一つであるトラッキングに着目し，この処理に要する時間（コスト）を推定し CPU の適切な動作周波数を設定することで低消費電力化を行う手法を提案する．トラッキング処理はある一定周期で一つの処理（フレーム処理）が繰り返される．本手法では，まずトラッキングの処理が，主に特徴点抽出処理とローカルマップ追跡処理の二つの処理から構成されていることに着目する．特徴点抽出処理の実行コストは抽出した特徴点数に強い相関があり，かつ最終的な抽出特徴点数は処理の早い段階で推定可能である．また，ローカルマップ追跡処理は，先行するフレームの当該処理コストと相関がある．このような性質に基づき，各処理の初期の段階で CPU の動作周波数を設定する．

さらに提案手法を，並列化した ORB-SLAM3 [8] に実装し，Nvidia の組み込み SoC ボードである Jetson Xavier NX 上で評価した結果について報告する．評価では Linux の標準的な電力制御ガバナである Performance 及び Ondemand と比較した．さらに，トラッキングのコスト推定手法に加重移動平均（Weighted Moving Average: WMA）を用いて電力制御を行った場合とも比較を行った．

以下本稿は，2. 節で関連するリアルタイム処理における低消費電力化技術を紹介し，4. 節で提案手法を説明する．5. 節で評価環境の説明と評価結果を述べ，最後に 6. 節でまとめる．

2. 関 連 研 究

Imes 等はソフトリアルタイム処理を対象に，DVFS の代わりにパワーキャップを設定することにより，制限時間を満たすのに十分な電力設定を行う手法を提案した [5].

Peters 等はマルチスレッドのゲームアプリケーションを対象に，各フレームの処理時間を推定して DVFS 制御を行うことにより低消費電力化を行う手法を提案した．本手法ではスレッド負荷の振る舞いとして，負荷が連続的に緩やかに変化しながら与えられるタイプと，周期的に発生するタイプがあることに着目し，前者を加重移動平均予測器（Weighted Moving Average: WMA），後者を自己相関予測器（Autocorrelation: ACR）で予測できるとした．そして，両者を組み合わせた Hybrid AMR による負荷の予測とこれを用いた DVFS 制御を提案した [4]．WMA 予測器では式 1 により過去のフレームの計算負荷から将来のフレームの計算負荷を推測する．

$$W(n+1) = \frac{\sum_{i=0}^{n-1}(N-i)\cdot W(n-i)}{\sum_{i=1}^{N} i} \tag{1}$$

ここで，$W(i)$ は i 番目のフレームの計算負荷，N は予測に使うフレームの数である．隣接フレーム間の時間的相関が高い場合はこの方法で良い精度で予測することができる．本稿で着目する ORB-SLAM3 のトラッキング処理は，負荷が連続的に変化するタイプの処理であり，WMA による予測が有効であると考えられる．

Khalufa 等は，Visual SLAM を対象とし，隣接フレーム間のカメラの移動度に基づき電力制御を行う手法を提案した [6]．提案手法を ORB-SLAM2 [9] と DSO [10] により評価した．

以上は動的な情報からフレームの実行コスト推定を行い電力制御を行う手法であるが，その一方でコンパイラによるプログラムの解析情報に基づき電力制御を行う手法も提案されている [1], [3]．動的な情報は負荷変動への追従性が高いが，実行時の推定オーバーヘッドを要する．これに対し，プログラムの静的解析情報を用いることができれば，このオーバーヘッドを削減可能と考えられる．

3. ORB-SLAM3 の処理の概要

3.1 ORB-SLAM3 の構成

図 1 に ORB-SLAM3 の構造を示す．図にあるように，ORB-SLAM3 ではトラッキング，ローカルマッピング，ループクロージングの 3 つの処理から構成されている．

トラッキングでは，カメラからの入力フレーム画像から FAST (Features from Accelerated Segment Test) [11] により，特徴点として地図作成に用いられる物体の輪郭や端点などを検出する．そして，前フレームとの比較によるカメラ位置推定や，ローカルマッピングで生成された地図内のカメラ位置補正（ローカルマップ追跡）を行い，最後に当該処理フレームが地図作成に必要なフレーム（キーフレーム）であるかどうかの選択を行う．

ローカルマッピングでは，フレーム内の特徴点とフレームを観測したカメラ位置からなるマップを作成する．この処理では，キーフレームの位置と内部の特徴点からなるマップポイントの生成・統合，およびローカルバンドル調整と呼ばれる地図上の点群を補正する最適化を行う．

ループクロージングでは，ローカルマッピングから挿入されたキーフレームを元に，マップ内で軌道がループになるかどうかを検出する．ループが検出された場合には，同じのフレーム位置の軌道を繋いでループにした上で，バンドル調整により地図上の全てのカメラ位置の補正を行う．

ORB-SLAM3 はこれら 3 つの処理は各々別々のスレッドとして実装されている．トラッキング処理は予め決められた周期でカメラ等のセンサから入力データを受け取り処理する．ローカルマッピングはトラッキング処理により算出されたキーフレームを受け取ると，その処理を開始する．さらにループクロージングはローカルマッピングで処理したキーフレームを受け取り処理を開始する．以上のように，トラッキング処理に連鎖してローカルマッピングとループクロージング処理がパイプライン

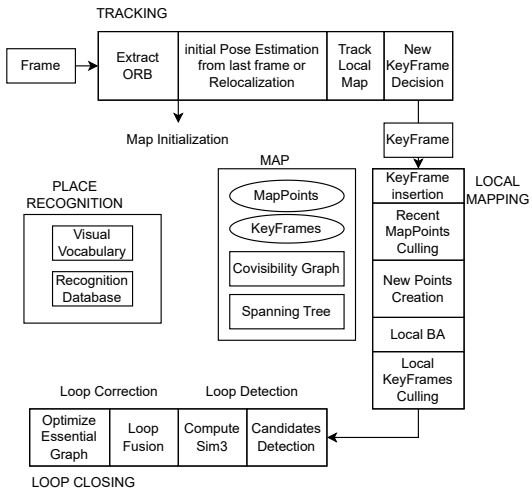

図 1　ORB-SLAM のシステム構成 [7], [8]

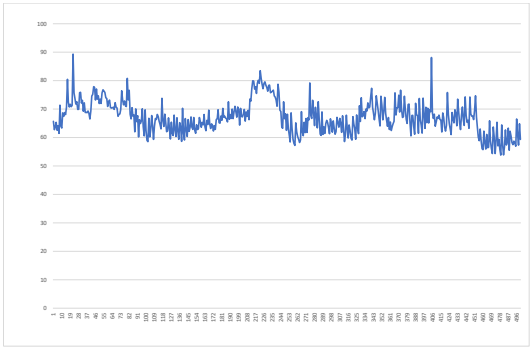

図 2　1 フレームあたりのトラッキング実行時間の推移.

的に処理される.

3.2　トラッキング処理の処理負荷の挙動

トラッキングの処理負荷変動の例を図 2 に示す．図は 5. 節で評価で用いた Nvidia Jetson Xavier Nx 上で KITTI データセット [12]0 番を実行したときの 501 番目のフレームから 1000 番目までのフレームのトラッキングの実行時間を OpenMP による並列化なし，最高周波数で固定の条件で計測したものである．

図 2 から各フレームの実行時間は一定せず，ある程度連続的ではあるがフレーム毎に大きく変化することがわかる．本図の場合，1 フレームあたりの実行時間の平均値は 67.0，標準偏差は 5.66 となる．すなわち，プログラム中に予め動作周波数を固定的に設定することでは動作時の変化に追従できず，適切な電力制御を行うことは極めて困難である．そのため，フレーム実行コストを正確に推定し，フレーム処理時間制限（デッドライン）を満たす動作周波数決定が重要となる．

以下では，トラッキング処理が主に前半の特徴点抽出処理と後半のローカルマップ追跡から構成されていることに着目し，各処理を実行コスト変化の振る舞いの観点から議論する．

3.2.1　特徴点抽出

特徴点抽出はトラッキング処理の実行時間の約 50% ほどを占める処理である．3.1 節で述べたように特徴点抽出処理では，

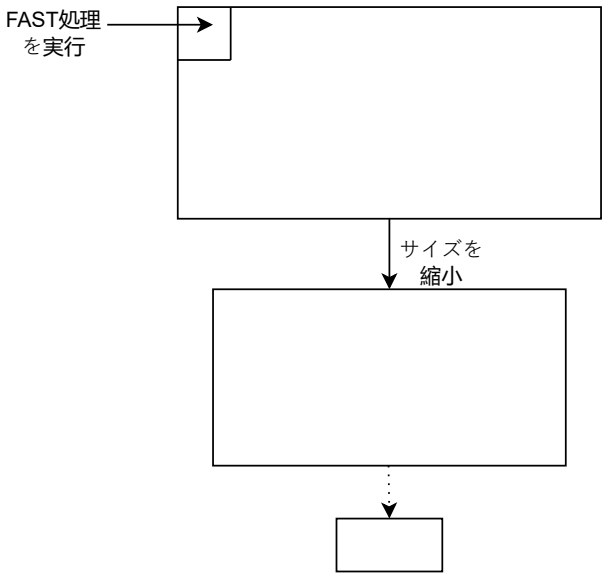

図 3　1 フレームに対する特徴点抽出の処理の流れ.

FAST アルゴリズム [13] によってフレームから特徴点を抽出する．この時，できるだけ高い精度で高速に特徴点を抽出するために，入力されたフレームのデータから大きさと解像度の異なる 8 層の画像データを重ねたピラミッド状のデータが生成され処理される（図 3）．次に，各層のデータを 35 × 35 の大きさのセルに分割した上で各セルに対して FAST を適用し，周辺領域に対して中心領域の画素値が大きい画素を特徴点として抽出する．この時 FAST では，ある画素の処理中にその画素が特徴点の条件を満たさないことがわかったら，後の処理はスキップされて次の画素の処理に進む．すなわち，フレーム中の特徴点の数が多い方が本処理に時間がかかる．

ここで，図 3 で示したように，特徴点抽出は大きさの異なる複数のデータに対して処理が行われることに着目する．これらのデータはオリジナルのフレームデータから得られたものであり，最初に処理した層の特徴点数が多い場合，後に処理する層の特徴点数も多いと予測できる．すなわち，特徴点抽出の早期の段階で本処理の実行時間が推定可能であると考えられる．このことを確認するために，最も小さい層で得られた特徴点数と特徴点抽出実行時間の関係を測定した（図 4）．本図の相関係数は 0.802 であり，最小の層で得られた特徴点数と特徴点抽出時間の相関は高く，これは実行時間の予測が可能であることを示している．

オリジナルの ORB-SLAM3 では，生成したピラミッド状データ構造の最も大きい層のデータから小さい方に向かって順に処理される．しかしながら，各層の処理には依存が無く任意の順番で実行可能である．そのため，最も小さい層の処理を最初に始めることで早期に推定コストを算出可能である．

3.2.2　ローカルマップ追跡

ローカルマップ処理はトラッキング処理の約 11% を占める処理である．ローカルマップ追跡では，ローカルマップを処理中のフレームに写像し，得られた特徴点と対応するポイントを探索する．この時，不要となった特徴点は削除され計算に用いら

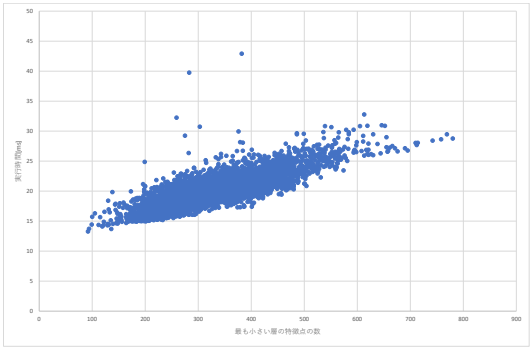

図 4　最も小さい層の特徴点の数と特徴点抽出の実行時間の関係.

れる．また，本追跡処理はローカルマップに含められたキーフレームに対して行われ，これらのキーフレームは時間的な局所性がある．そのため，ローカルマップ追跡の処理時間は 3.2.1 節で得られた特徴点数よりもむしろ，直近の数フレーム分の当該処理の処理時間から推定可能と考えられる．

4.　提案する周波数制御手法

本節で提案する ORB-SLAM3 のトラッキング処理コスト推定による動作周波数制御手法を説明する．なお，本提案ではソフトリアルタイム制御を目指す．

3.2 節で述べたように，トラッキング処理は実行コストの観点から主に特徴点抽出処理とローカルマップ追跡処理の二つの処理から構成される．そこで，周波数決定のタイミングを二段階に分け，これらの処理を行う際に各処理の推定実行コストを算出し動作周波数を決定・設定する．本稿では，トラッキングの処理周期を 100ms と設定する．これはオリジナルの ORB-SLAM3 のサンプル実行の設定と同一である．その上で，動作周波数制御や並列実行時の同期のオーバーヘッドを加味して，トラッキング 1 フレームの目標実行時間を 80ms とし，実行時間がこの値を超えないように動作周波数を決定する．この時，あるタスクの実行時間は周波数に反比例という仮定の下で推定実行コストから動作周波数を求める．

第一段階目として特徴点抽出処理の実行コストを推定する．3.2.1 節で述べたように，ピラミッド状データ構造の最小の層のデータで抽出した特徴点数と特徴点抽出処理の実行コストは相関が高い．そこで，抽出特徴点数から特徴点処理実行コストを導出する式を，トラッキング処理の実行プロファイルから線形回帰により求めた．プロファイル取得のため，図 2 と同様に KTTI データセットの 0 番を単一スレッドで実行した．得られた線形回帰式を式 2 に示す．ここで，t は最高周波数時の特徴点抽出処理の実行時間 [ms]，n は最も小さい層で抽出した特徴点の数である．

$$t = 0.0232n + 11.7 \tag{2}$$

なお，オリジナルの ORB-SLAM3 では，ピラミッド状データ構造の各層の処理は，大きい層から小さい層の順番で行われるが，3.2.1 節で述べたように層の処理間に依存は無い．そのため，本稿ではまず最小の層の処理を行いその結果によりコスト推定を行った後，残りの層の処理をオリジナルと同じ順番で行った．

表 1　Jetson Xavier Nx の CPU の諸元

CPU コア	NVIDIA Carmel ARM®v8.2 64-bit
コアの数	6
最大周波数	1.91GHz
L1d Cache	64KB
L2 Cache	2048KB
L3 Cache	4096KB

第二段階目のローカルマップ追跡処理は，3.2.2 節で述べたように実行時間の時間的局所性が高い．そのため，実行コスト推定は式 1 の WMA により算出する．式中のパラメータ N は，本稿の評価では 10 フレームとした．

各段階共に，以下の式 3 により設定する動作周波数を求める．ここで，f は求める CPU の動作周波数，f_{max} は CPU に設定可能な最大動作周波数，T_e はフレーム処理の開示時刻からコスト推定時点までの経過時間，T_p は得られた推定コスト，をそれぞれ表す．

$$f = \frac{T_p - T_e}{80 - T_e} f_{max} \tag{3}$$

5.　評　　価

5.1　評 価 環 境

4. 節で述べた提案手法を並列化した ORB-SLAM3 [8] に実装し，Nvidia の組み込み SoC ボード Jetson Xavier NX で評価した．Jetson Xavier NX の CPU の諸元を表 1 に示す．本 SoC は GPU を持つが，本評価では CPU コアのみを使用した．

電力制御機構に関しては，本 SoC は DVFS，クロックゲーティング，パワーゲーティングが可能である．ただし，DVFS は全てのコアの動作周波数を一括で変更する必要があり，各コア独立の制御は不可能である．

CPU コアの電力制御は Linux の周波数ガバナを用いる．Linux の周波数ガバナとして本評価では Performance，Ondemand，Userspace の 3 種類を用いた．それぞれ，最大動作周波数での動作，OS による実行時負荷に基づく動作周波数自動制御，及びユーザプログラムからの設定値に基づいた動作周波数制御となる．

本評価で用いた ORB-SLAM3 は，トラッキング処理が定められた制限時間（本評価では 100ms）未満で 1 フレームの処理を終了すると，sleep により残り時間は当該スレッドが待ち状態になる．Ondemand ガバナではこの時に周波数制御あるいはクロックゲーティングを行う．

Userspace ガバナで設定可能な動作周波数とその時の動作電圧を表 2 に示す．本 SoC では CPU コアの動作周波数を 115MHz まで設定可能だが，動作電圧は 1.34GHz の 681mV が下限でありこれよりも低い値にはならない．

提案手法を実装した ORB-SLAM3 は Userspace ガバナで実行

表 2　Jetson Xavier Nx のサポートする CPU の周波数と対応する電圧

周波数 [GHz]	電圧 [mV]
1.34	681
1.42	690
1.50	730
1.57	730
1.65	752
1.73	791
1.80	791
1.88	810
1.91	820

し，4. で述べた手法に従い動作周波数を設定した．オリジナルの ORB-SLAM3 は 3. 1 で述べたようにトラッキング，ローカルマッピング，ループクロージングの各処理に 1 スレッドずつ使用している．今回評価した並列化版で並列処理実行した場合，トラッキングを 2 スレッド，ローカルマッピングを 3 スレッドで処理し，ループクロージングと合わせて計 6 スレッドで実行した [8].

評価には，ステレオカメラ版であり，車上のカメラフレーム画像から構成される KITTI データセット [12] を使用した．使用したデータセットは 0 番から 2 番だが，いずれの場合も提案手法の特徴点抽出処理コスト推定は 0 番のプロファイル結果から得られた線形回帰式を用いた．KITTI0 番は 4541 枚，1 番は 1101 枚，2 番は 4661 枚のフレームから構成される．比較対象として，Performance ガバナ，Ondemand ガバナ，及び予測機として WMA を使用したものも評価した．WMA 使用の場合，式 1 のパラメータ N は 10 フレームとした．

5.2　評 価 結 果

評価結果を表 3 に示す．KTTI 番号 0，１，2 と並列化有無の組み合わせを提案手法，Perforamcne ガバナ，Ondemand ガバナ，及び WMA 予測器で評価した．評価項目として，デッドライン（100ms）超過フレーム数と消費電力に加えて，トラッキング平均時間及び絶対軌道誤差（Absolute Trajectory Error: ATE）を計測した．

表より，提案手法を Performance ガバナと比較すると消費電力をそれぞれ削減しており，並列化無しの場合で KTTI0 番，1 番，2 番でそれぞれそれぞれ 73.4%，72.3%，及び 71.8% の消費電力で実行できていることがわかる．その一方でデッドライン超過フレーム数は若干増加してしまい，各 KTTI 番号 0 番では 5 から 10，KITTI2 番では 6 から 7 となっている．KTTI 番号 1 では逆に 4 から 2 に減少している．Performance ガバナは最大周波数での動作なので，基本的に本ガバナでの超過フレーム数が評価に使用した Jetson Xavier NX での最小値となる．

また，提案手法を Ondemand と比較すると，デッドライン超過フレーム数と消費電力を共に削減できていることがわかる．並列化無しの場合の各 KTTI 番号で，順に消費電力では 89.1%，86.6%，87.9% となり，超過フレーム数は 1.5%，0.8%，1.2% となった．

以上のように KTTI0 番，1 番，２番のいずれに対しても性能を維持しつつ消費電力を削減できており，KTTI0 番のプロファイル結果から得られた特徴点抽出処理実行コスト算出の回帰式が他のデータセットでも有効であることが確認できた．

表 3　評価結果まとめ

KITTI 番号	並列化の有無	周波数制御方式	トラッキング平均時間 [ms]	デッドライン超過フレーム数	消費電力 [mW]	ATE [m]
0	無し	Performance	68.7	5	2382	1.34
1	無し	Performance	79.9	4	2804	13.68
2	無し	Performance	68.0	6	2407	5.88
0	有り	Performance	57.9	6	2508	1.34
1	有り	Performance	71.0	4	2976	14.11
2	有り	Performance	56.2	6	2537	5.42
0	無し	Ondemand	87.7	665	1963	1.22
1	無し	Ondemand	89.6	241	2342	13.34
2	無し	Ondemand	87.4	603	1969	6.77
0	有り	Ondemand	79.6	82	1773	1.23
1	有り	Ondemand	88.1	162	2121	13.54
2	有り	Ondemand	78.2	21	1777	5.78
0	無し	WMA	74.9	9	1909	1.39
1	無し	WMA	80.5	5	2197	13.45
2	無し	WMA	74.0	5	1912	5.94
0	有り	WMA	73.6	11	1761	1.26
1	有り	WMA	77.0	11	2261	13.69
2	有り	WMA	73.5	5	1704	6.27
0	無し	提案手法	79.8	10	1750	1.25
1	無し	提案手法	82.7	2	2028	12.99
2	無し	提案手法	80.7	7	1730	6.18
0	有り	提案手法	78.8	16	1584	1.28
1	有り	提案手法	81.4	4	2025	14.41
2	有り	提案手法	79.2	19	1560	6.16

WMA と比較すると，並列化無しの場合に提案手法の消費電力は各 KTTI 番号に対して 91.7%，92.3%，90.5% となっており，一方でデッドライン超過フレーム数はほぼ同じという結果を得ている．

並列化の有無に着目すると，提案手法の並列化有りは並列化無しに対して各 KTTI 番号に対して 90.5%，99.9%，90.2% の消費電力で実行できている．並列処理による処理の高速化により，並列化無しの場合に対してより低い動作周波数を設定可能であり，結果としてさらなる消費電力削減を得られた．一方で，平均実行時間は維持しているものの超過フレーム数は増加してしまっている．これは，並列化によってたくさんのスレッドが生成され，競合が発生するためであると考えられる．

5.3　フレームごとのトラッキング時間

OpenMP による並列化なしで提案手法による周波数制御を行った場合の KITTI0 番の 501 フレーム目から 1000 フレーム目までの実行時間を図 5 に示す．

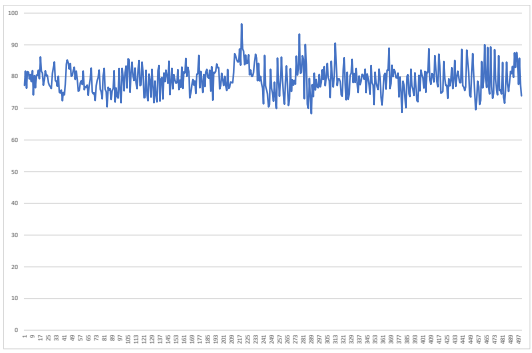

図 5　提案手法を適用した場合の 1 フレームあたりのトラッキング実行時間の推移.

提案手法により実行時間の平均は 79.0，標準偏差は 4.34 であった．図 2 に示す最高周波数時と比較すると，平均は大きくなりほぼ目標性能値である 80ms と同一になり，また標準偏差は小さくなり計算負荷に応じて周波数を制御することに成功していることがわかる．

6. ま と め

本稿では，ORB-SLAM3 のトラッキング時間がフレームによって異なる点に着目してフレームの計算負荷に応じた周波数制御を行うことで低消費電力化を実現する手法を提案した．

提案手法では，トラッキングの前半部である特徴点抽出処理では，本処理の実行時間と，抽出される特徴点の数との間に相関が見られたので最も小さい層での抽出される特徴点の数から特徴点抽出の計算負荷を線形回帰によって推定した．また，トラッキングの後半部であるローカルマップ追跡処理に関しては，本処理の実行時間に時間的局所性があることから，加重移動平均予測器による推定を行った．そして，上記 2 処理より得られたトラッキング処理の推定コストに基づき CPU コアの適切な動作周波数を設定した．

提案手法を Nvidia の組み込み SoC ボード Jetson Xavier NX 上で評価した結果， Linux の Performance ガバナと比較するとデッドライン超過フレーム数は KITTI2 番では 6 から 7 のように増えてしまうが，消費電力は，71.8% にまで削減できた．また，WMA 予測器をトラッキング全体の時間に適用した場合と比較すると，デッドライン超過フレーム数はほぼ同じで，消費電力は KITTI2 番では 90.5% に減らせた．提案手法によってデッドライン超過フレーム数を十分に抑えた上で ORB-SLAM3 の低消費電力化が達成できた．

謝辞　本研究の成果の一部は国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務 JPNP16007 の結果得られたものです．

文　献

[1] Jun Shirako, Munehiro Yoshida, Naoto Oshiyama, Yasutaka Wada, Hirofumi Nakano, Hiroaki Shikano, Keiji Kimura, and Hironori Kasahara. Performance evaluation of compiler controlled power saving scheme. In *Lecture Notes in Computer Science, Springer*, Vol. 4759, pp. 480–493, January 2008.

[2] Sabrina M. Neuman, Jason E. Miller, Daniel Sanchez, and Srinivas Devadas. Using application-level thread progress information to manage power and performance. In *2017 IEEE International Conference on Computer Design (ICCD)*, pp. 501–508, 2017.

[3] Tomohiro Hirano, Hideo Yamamoto, Shuhei Iizuka, Kohei Muto, Takashi Goto, Tamami Wake, Hiroki Mikami, Moriyuki Takamura, Keiji Kimura, and Hironori Kasahara. Evaluation of automatic power reduction with oscar compiler on intel haswell and arm cortex-a9 multicores. September 2014.

[4] Nadja Peters, Dominik Füß, Sangyoung Park, and Samarjit Chakraborty. Frame-based and thread-based power management for mobile games on hmp platforms. In *2016 IEEE 34th International Conference on Computer Design (ICCD)*, pp. 169–176, 2016.

[5] Connor Imes, Huazhe Zhang, Kevin Zhao, and Henry Hoffmann. Copper: Soft real-time application performance using hardware power capping. In *2019 IEEE International Conference on Autonomic Computing (ICAC)*, pp. 31–41, 2019.

[6] Abdullah Khalufa, Graham Riley, and Mikel Lujan. Poster: Runtime adaptations for energy-efficient vslam. In *2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 475–476, 2019.

[7] Carlos Campos, Richard Elvira, Juan J. Gomez Rodriguez, Jose M.M Montiel, and Juan D. Tardos. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. *IEEE Transactions on Robotics*, Vol. 37, No. 6, p. 1874–1890, 2021.

[8] 山本一貴, 長ヶ部拓吾, 小池穂乃花, 川角冬馬, 藤田一輝, 北村俊明, 川島慧大, 納富昭, 木村貞弘, 木村啓二, 笠原博徳. Orb-slam3 のローカルマッピングの並列化とコア割り当て手法の提案. 電子情報通信学会技術報告, 第 121 巻, pp. 79–84, March 2022.

[9] Raúl Mur-Artal and Juan D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, Vol. 33, No. 5, pp. 1255–1262, 2017.

[10] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 40, No. 3, pp. 611–625, 2018.

[11] OpenCV: FAST Algorithm for Corner Detection. `https://docs.opencv.org/4.5.2/df/d0c/tutorial_py_fast.html`. (Accessed on 01/07/2022).

[12] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, Vol. 32, No. 11, pp. 1231–1237, 2013.

[13] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pp. 430–443, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.