イベントカメラを用いた人物姿勢推定結果の更新による レイテンシ補償と精度向上

大武 一平[1,a)]　北野 和哉[1]　櫛田 貴弘[1]　藤村 友貴[1]
前島 謙宣[2]　久保 尋之[3]　舩冨 卓哉[1]　向川 康博[1]

概要

従来の姿勢推定をオンラインプロセスで実施すると，姿勢推定に係る処理時間により画像が取得された時刻と姿勢推定結果が出力される時刻の時間差が姿勢推定の精度低下を招き，リアルタイム性の低下につながる．本研究では，イベントカメラが持つ低レイテンシ性と時間分解能の高さを活かし，姿勢推定結果が最新の姿勢と一致するよう更新する手法を提案する．また，イベントカメラの非同期性を活かし，人の動きの大きさに応じた適応的な処理を行うことで，姿勢推定の精度が向上することを示す．

1. はじめに

現在，2D アバターを動かし動画を配信するエンタテイメントコンテンツや人の動きによってマッピング内容を変化させる動的プロジェクションマッピングといった，人との間でリアルタイム性のあるインタフェースが求められるアプリケーションが普及しつつある．これらのアプリケーションは人の速い動きに対して追従することが必要不可欠である．しかし，従来の人の姿勢推定技術をオンラインプロセスで用いると，姿勢推定器の処理時間によるレイテンシがリアルタイム性を低下させる．特に人の動きが速い場合，姿勢推定結果が得られた時点では既に異なる姿勢となっており，姿勢推定精度の低下を招く．

本稿では，輝度値の変化を低レイテンシかつ高い時間分解能で出力するイベントカメラ [2] を用いて，既存の姿勢推定出力を高速に更新することで，レイテンシを補償し姿勢推定精度の向上を図る．この際，姿勢推定の処理中に取得されるイベントデータを一括で処理して姿勢を更新するのではなく，動きの大きさに対して適応的に姿勢の変化量を多段階で導出するすることで，姿勢推定の更新精度が向上することを示す．

1　奈良先端科学技術大学院大学
2　株式会社オー・エル・エム・デジタル，株式会社 IMAGICA GROUP
3　千葉大学
a)　otake.ippei.oj2@is.naist.jp

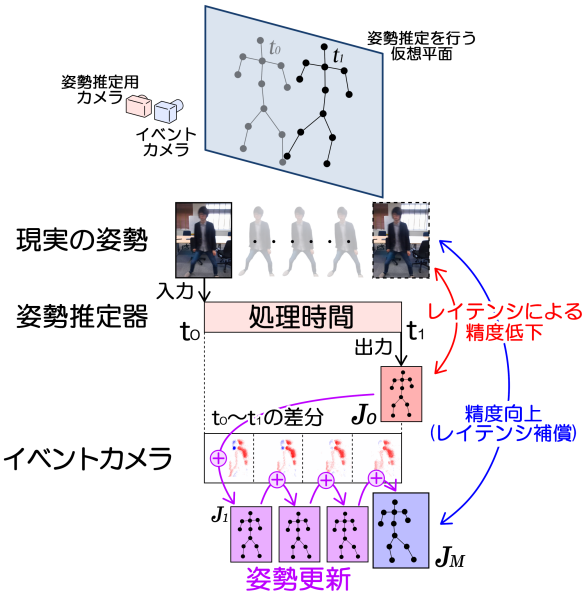

図 1　提案手法では，姿勢推定用フレームカメラと別にイベントカメラでシーン情報を取得する．姿勢推定処理のレイテンシによる推定精度の低下を補償するため，イベントデータを用いた姿勢変化量導出を行い，関節位置の更新を行う．

2. 提案手法

2.1 問題設定

今回我々が取り組むタスクを図 1 に示す．従来のフレームカメラで人物画像を撮影した時刻を t_0 とする．その画像に対する姿勢推定結果が出力される時刻を t_1 とする．本研究では t_0 での画像に対する姿勢推定結果を t_1 での姿勢に更新するレイテンシ補償に取り組む．この更新は，t_0〜t_1 に取得されたイベントデータを用いて行う．提案手法では，イベントデータを時間方向の姿勢変化量と捉え，ある時刻の関節位置に対してその姿勢変化量を足し合わせることで，関節位置を更新するアプローチをとる．

2.2 姿勢推定処理時間中の姿勢変化量導出

提案手法では，t_0〜t_1 に取得されたイベントデータを一括処理して姿勢変化量を推定するのではなく，一定量のイ

ベントデータが得られるごとに順次，処理時間由来の姿勢変化量を導出する処理を行う．本稿では姿勢変化量として Dense Optical Flow を用い，姿勢推定が完了すると同時に姿勢更新を行うことでレイテンシ補償を達成する．

姿勢変化量の導出は以下のモジュールから構成される．

(1) イベントデータを積算することでイベント積算画像を作成する，イベント積算処理.
(2) 一定数ごとにイベントデータをバッファリングする，適応的イベントバッファリング処理.
(3) バッファリングされたイベントデータから，更新に有用なイベントのみを抽出する，イベント数制限処理.
(4) 2 枚のイベント積算画像から関節位置の更新量を推定する，Dense Optical Flow 導出処理.

2.2.1 イベント積算処理

本処理ではイベントデータをイベント積算画像に変換する．イベントデータ $\boldsymbol{E}_{n:m} = \{E_i \mid n \leq i < m\}$ はイベントカメラから以下の形式で出力される．

$$E_i = (x, y, p, t) \tag{1}$$

ここで n, m はイベントデータの系列番号，t は時刻で，p はピクセルの輝度値がどう変化したかを表すバイナリ値である．イベント積算画像 $I_{n:m}$ はこのイベントデータ $\boldsymbol{E}_{n:m}$ を以下の形で 3 種類の輝度値を持つ 1 枚の画像に整形したものである．

$$I_{n:m}[x_0, y_0] = \begin{cases} 255 & (x_0, y_0, \mathrm{Pos}, t) \in \boldsymbol{E}_{n:m} \\ 127 & (x_0, y_0, p, t) \notin \boldsymbol{E}_{n:m} \\ 0 & (x_0, y_0, \mathrm{Neg}, t) \in \boldsymbol{E}_{n:m} \end{cases} \tag{2}$$

2.2.2 適応的イベントバッファリング処理

本処理はイベント積算画像の作成をある一定のイベント数ごとに行うことで，動きの大きさに応じた適応的な処理を行う．人物の動きが速い場合には一回で姿勢変化量を導出することは困難であるため，多段階でイベントデータから姿勢変化量を導出すことで精度良く姿勢の更新を行う．多段階化に際して，時間軸に対して等間隔にイベントデータを分割すると，単位時間あたりのイベント数が少ない場合には動きに関する情報のないイベント積算画像を作成してしまう．逆に単位時間あたりのイベント数が多い場合には，1 枚のイベント積算画像が持つ動きに関する情報が過多となり，後段の更新量推定に悪影響を及ぼす．イベント数に基づいて多段階化を行うことで，推定に用いるイベント数が毎回等しく，入力される情報量が均一になる他，関節位置の更新が不要な静的なシーンでは計算を行わない適応的なアプローチとなる．

1 枚のイベント積算画像を作成するのにバッファリングするイベント数を N_b とすると，m 枚目のイベント積算画像は $\boldsymbol{E}_{mN_b:(m+1)N_b}$ から作成することになる．

2.2.3 イベント数制限処理

本処理は，バッファリングした N_b 個のイベントデータから最新の N_l 個のイベントデータ $\boldsymbol{E}_{(m+1)N_b-N_l:(m+1)N_b}$ のみを取り出すことで，関節位置の更新量推定に有用な差分情報を得る．

イベント数の制限がない場合とある場合の違いを図 3 に示す．制限なし (a) の場合，$\boldsymbol{E}_{mN_b:(m+1)N_b}$ の全てを用いて生成したイベント積算画像は，動きに応じて連続的な軌跡を画像上に描く．これに対し，直近のイベントデータのみを用いてイベント積算画像を生成する (b) では，動きがあった部分の輪郭に相当する画像が得られる．(b) のほうがより関節位置の更新量として，動きによる差分を正確に捉えられることが期待される．また，使用するイベント数が減少することによる計算量抑制も期待できる．

2.2.4 Dense Optical Flow 導出処理

本処理は，時間的に連続した 2 枚の イベント積算画像から，関節位置の更新量の推定値として，Dense Optical Flow を求める．本稿では画像を階層化することでピクセルごとに二次元速度 f を持つ場である Dense Optical Flow を比較的正確に求められる Farneback 法を用いた [1].

イベントデータ $\boldsymbol{E}_{mN_b-N_l:mN_b}$ と $\boldsymbol{E}_{(m+1)N_b-N_l:(m+1)N_b}$ から生成された 2 枚のイベント積算画像である $I_{mN_b-N_l:mN_b}$ と $I_{(m+1)N_b-N_l:(m+1)N_b}$ から求めた Dense Optical Flow を $F_{m:m+1}$ と表記する．

$$F_{m:m+1} = \{f_{m:m+1}[x, y] = (u, v) \mid x, y\} \tag{3}$$

2.3 姿勢更新によるレイテンシ補償

姿勢推定に入力する画像が取得された t_0 直後から，前項で述べた姿勢変化量導出処理を繰り返し行う．姿勢推定処理が完了した t_1 の段階で，M 個の Dense Optical Flow が得られるものとする．出力された姿勢推定結果 $\boldsymbol{J}_0$ である t_0 における関節位置を $\boldsymbol{J}_0 = \left\{J_0^k = (x_k, y_k) \mid k \in K\right\}$ とする．ここで k は関節番号である．イベントデータをもとに得られた Dense Optical Flow を用いて，推定された関節位置を更新する．この更新は漸化的に次式で行う．

$$J_{m+1}^k = J_m^k + F_{m:m+1}[x_k, y_k] \, (m = 0, \cdots, M-1) \tag{4}$$

これを M 回繰り返すことにより，レイテンシ補償後の関節位置 $\boldsymbol{J}_M$ を得ることができる．

姿勢変化量の導出は姿勢推定処理と並列して実行する．一方，関節位置の更新処理は姿勢推定出力が得られたと同時に実行することで，更新処理に係るレイテンシを補償する．

3. 実験設定

3.1 実験環境

フレームカメラとして Azure Kinect，イベントカメラに

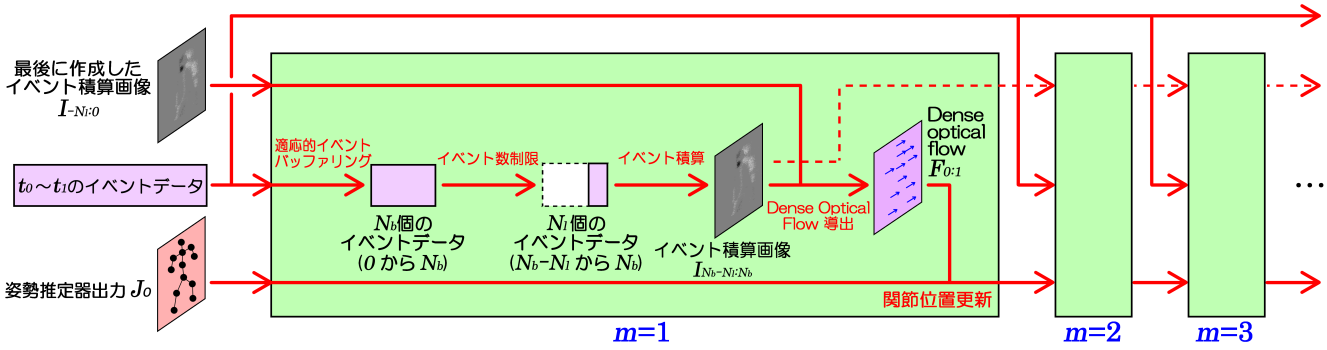

図 2 提案手法のアーキテクチャ．レイテンシ補償処理は姿勢推定出力が得られると即時に実行され，姿勢変化量の導出はイベント取得と並行して実行される．

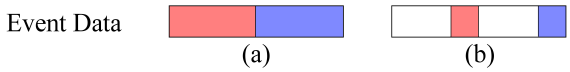

図 3 イベント数制限処理の概略．差分を捉える幅を制限する．

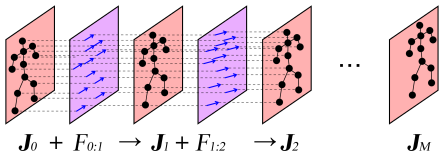

図 4 関節位置更新の概略，入力の関節位置座標を Dense Optical Flow 上で参照し当該座標の Optical Flow を入力座標に足し合わせて出力座標とする．これを繰り返す．

は Prophesee Gen3 を用いた．また，ベースラインの姿勢推定器として Azure Kinect Body Tracking を用いた [3]．通常，フレームカメラによる 2 次元の姿勢推定結果の補正を別のカメラで行う場合，カメラの位置を完全に同じにすることができず厳密な位置合わせが不可能となる．今回はこの問題を，Azure Kinect Body Tracking の 3 次元姿勢推定結果をイベントカメラ視点の 2 次元姿勢へ変換することで回避している．また，イベントデータのバッファリングは $N_b = 50,000$ 個ごとに行い，収集したイベントデータからイベント数制限処理を以て $N_l = 10,000$ 個を抽出した．

3.2 データセット

本手法はオンライン処理を目的としているが，各種処理の有効性を検証するため，本実験ではオンラインで用いる環境でデータセットを収録し，オフラインで各種処理を実行し評価を行った．データセットとして，被験者が撮影範囲で左右に動く反復横跳びのような動作を収録した．収集したデータは以下の 3 つから構成される．

(1) Azure Kinect Body Tracking による姿勢推定出力
(2) (1) の出力を取得した際の現実の参考画像
(3) (1) の処理時間中に取得したイベントデータ

姿勢推定器に関しては，姿勢推定処理が完了した瞬間に次の姿勢推定入力であるフレーム画像を取得するようフレームカメラと同期させている．具体的にはフレームカメラが 30fps でシーンをサンプリングする一方で，姿勢推定器も 1 フレーム分のレイテンシを以て 30fps で姿勢推定出力を行なった際に得られるストリームをデータセット化している．これにより，姿勢推定処理が完了した瞬間に次フレームの画像を取得しているため，姿勢推定処理が完了した瞬間の正解関節位置は次フレームの画像から推定した関節位置と一致していると考えることができる．

3.3 評価指標

(1) に対するレイテンシ補償処理の精度評価として，次フレームの (1) を正解関節位置とし，mean per joint position error（MPJPE），probability of correct keypoint（PCK）の 2 種類の指標を評価した．MPJPE は各関節の出力値と正解値の関節平均誤差である．PCK は出力値が正解から代表長さ内にある割合で，本稿では代表長さを頭部—首の距離とする PCKh を用いた．またベースラインには入力に用いた (1) をそのまま次の (1) と比較したものを用いた．

4. 実験および結果と考察

4.1 レイテンシ補償処理の精度評価

定量的な評価結果を図 6 に示す．図 6(a) より，本プロセスを行わない場合は動作が激しくなるにつれて MPJPE が大きくなり，ベースラインでは動作が激しくなるにつれてレイテンシが増大した．一方，提案手法を用いると動作が激しい場合でも MPJPE を一定以下に抑えられており，動作が激しいときの追従性が著しく向上した．一方で，動作が小さい 0, 17 フレーム付近では MPJPE の改善は見られなかった．これは，イベントデータを N_b 個で区切りイベント積算画像にしているため，イベント数が減少する微小な動きには追従できていないと考えられる．また図 6(b) より，提案手法を用いると元の姿勢推定出力に比べてプロットが左方向に大幅にシフトしており，〜PCKh@0.5 で正解率は 2 倍程度向上している．以上から，提案手法は誤差を一定以下に抑える効果があると言える．

また提案手法の平均処理時間を表 1 に示した．提案手法の姿勢変化量導出にかかる処理時間が 6.79[ms] である一方，本実験環境で姿勢推定出力は最大 30fps で出力される．

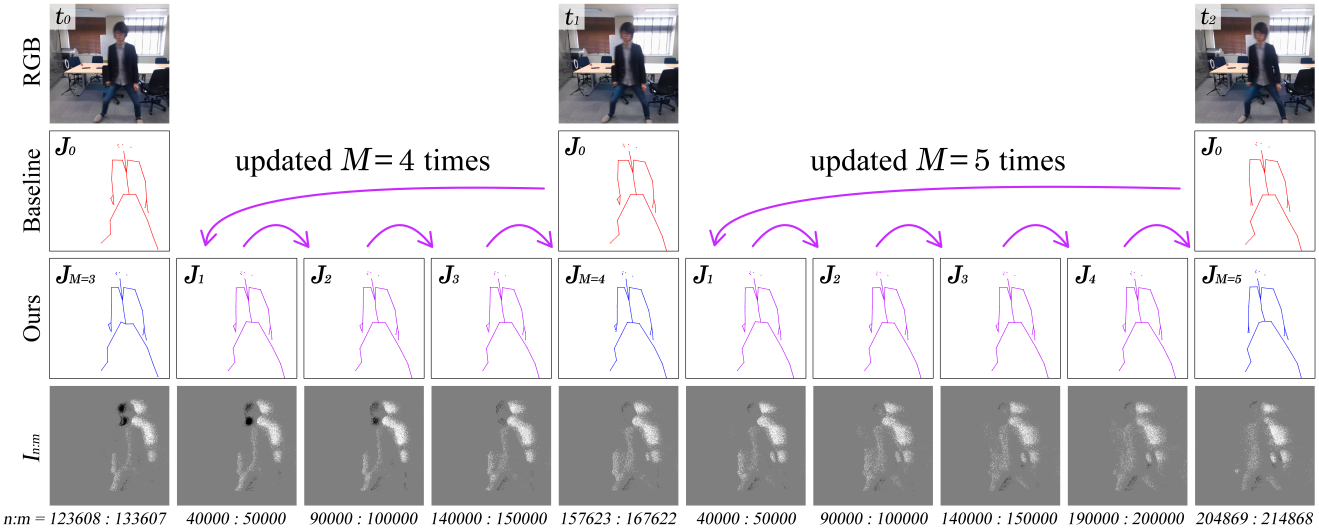

図 5 定性評価．レイテンシ補償処理を赤色，関節位置の更新結果を青色で示し，更新処理に使用したイベント積算画像を下段に示した．動きに適応的な処理により前後半でフレーム補間による出力回数が異なる．

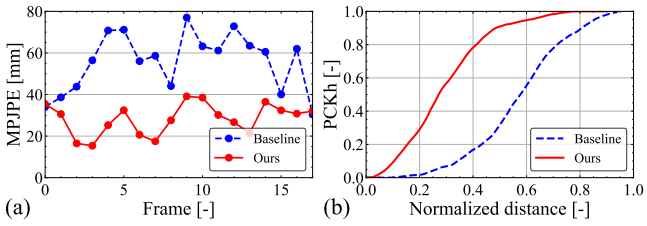

図 6 レイテンシ補償処理の定量評価．連続したビデオフレームにおける MPJPE の推移を (a) に，全フレーム平均の PCKh を (b) に示した．

表 1 提案手法の平均処理時間．

モジュール	処理時間 [ms]
イベント積算処理	3.34
Dense Optical Flow 導出	3.45

表 2 各モジュールの有効性検討．

適応的 Buffering	イベント数制限	イベント画像 3 値化	MPJPE [mm](↓)	PCKh @0.5(↑)
	Baseline		55.8	0.299
			50.8	0.458
		✓	48.1	0.472
	✓	✓	57.7	0.264
✓		✓	29.1	0.875
✓	✓		28.5	0.910
✓	✓	✓	**28.3**	**0.917**

これらから，本プロセスは姿勢推定出力を待つ間に 5 回程度までフレーム補間が可能である．

4.2 レイテンシ補償処理の定性評価

提案手法による姿勢の更新結果を図 5 に青いボーンで示す．前半に比べ後半は動きが大きく，姿勢の更新回数が適応的に増加している．

4.3 Ablation Study

姿勢変化量の導出における各モジュールの有効性を検証した．適応的イベントバッファリングを行わない場合として，N_b を姿勢推定の処理時間中のイベントを一括で処理し，一段階でレイテンシ補償処理を実施した．イベント数制限処理を行わない場合として，$N_l = N_b$ とし，バッファリングしたイベントデータ全てを用いてイベント積算処理を行った．イベント画像の 3 値化を行わない場合として，イベントを検知したピクセルを 255，それ以外を 0 として 2 値でイベント積算処理を行った．

表 2 に示した結果から各モジュールでの処理の有効性が示された．特に適応的イベントバッファリングを行わずイベント数制限を行った場合には，時間軸方向に長いデータの前後の一部データしか用いず時間的なつながりが極端に薄くなり，性能を大きく低下させる結果が確認された．

5. 結論

本稿では，姿勢推定出力に対してイベントカメラの低レイテンシ性と時間分解能の高さを活かした高速な更新処理を行い，姿勢推定のレイテンシ補償と推定精度向上を図った．提案手法が処理時間由来の残差を補正することで，レイテンシを補償し，姿勢推定の精度が向上することを実験により示した．また，イベント数に応じたフレーム補間や利用するイベントデータの選択を行うことが，姿勢推定精度の向上に有効であることを示した．

本実験ではオフライン処理での処理時間を以って有効性を示したが，オンライン実行での検証は今後の課題である．

謝辞
本研究の一部は JST さきがけ JPMJPR2025，JSPS 科研費 JP23K16902 の支援を受けた．

参考文献

[1] Farnebäck, G.: Two-Frame Motion Estimation Based on Polynomial Expansion, *Image Analysis* (Bigun, J. and Gustavsson, T., eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 363–370 (2003).

[2] Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A. J., Conradt, J., Daniilidis, K. and Scaramuzza, D.: Event-Based Vision: A Survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 44, No. 1, pp. 154–180 (2022).

[3] Liu, Z.: 3D Skeletal Tracking on Azure Kinect – Azure Kinect Body Tracking SDK, *3D Computer Vision in Medical Environments in conjunction with CVPR* (2019).