第13回人工知能学会全国大会（JSAI'99）, pp.105-108 (1999)

映像からの動き物体の分離・追跡の実時間処理

Real-time segmentation and tracking of moving objects from a video sequence.

興梠 正克 † 蔵田 武志 ‡ 村岡 洋一 †
Masakatsu Kourogi† Takeshi Kurata‡ Yoichi Muraoka†

† 早稲田大学 理工学部 ‡ 電子技術総合研究所
†Waseda University ‡Electrotechnical Laboratory

Abstract: In this paper, we propose a fast and robust method of segmentation and tracking of moving objects from a video sequence taken from a hand-held camera. Firstly, the proposed method estimates global affine motion parameters between two successive frames. Secondly, it selects pixels on which pixel-wise matching by the estimated motion fails, and selected pixels are merged into several non-overlapping regions. Thirdly, votes for the pixels of the segmented regions are accumulated over time. Finally, the method selects and tracks the regions consisting of pixels with sufficiently high accumulated vote counts. We implement the method as a software for general PC. Experimental results show that this method can estimate global motion, segment and track independently moving objects at 10-12 frame/sec.

1 はじめに

映像から動き情報に基づいて物体を分離・追跡することは，映像を扱う応用例にとって有用な要素技術である．例えば，映像中に提示された物体の認識や形状復元においては，処理の対象を絞るため，その対象が背景から前もって分離されていることが望ましい．また，映像中の背景と物体を分離して，そのレイヤ表現を構築することは，映像の知的符号化やビデオモザイキング [2][4] などの応用例に有用である．特に，その実時間処理はリアルタイム応答性が要求される応用例にとって必要不可欠である．近年ではモバイル PC とカメラを用いた多くの応用例が提案，検討されており，その前処理として，特殊な専用ハードウェアなしで動き物体の分離と追跡を実時間処理できることが望まれている．

本研究では，視差の影響が小さい状況下でフリーハンド撮影された視点非固定のカメラ映像を対象として，動き物体の分離と追跡の実時間処理を PC のソフトウェアにより実現することを目的とする．

映像から動き領域を分離・追跡する手法については，多くの従来手法が提案されてきた [3][4][5]．しかしながら，処理の計算コストが高く，汎用的な PC のソフトウェアによる実時間処理は難しい．また，処理の実時間性を考慮した従来研究としては，固定されたカメラの映像を対象として，背景差分に基づいて動き物体を分離する手法が提案されているが，その適用範囲は制限される．

我々は，フレーム間のグローバルなアフィン動きパラメータ推定を実時間処理できる手法を提案している [1]．本研究では，この手法を用いて得られる動きパラメータに基づいて，フレーム間で画素単位マッチングを行う．マッチングに失敗する画素とその周辺領域を動き物体の候補とみなして，これらの候補領域を併合して動き領域を分離する．次に，各フレーム間で分離された動き領域の画素位置に投票する．時系列方向に沿って高い投票数を得た領域を信頼できる動き領域とみなして出力する．

本論文の構成は以下の通りである．2-3 では，動き領域を分離・追跡する提案手法について述べる．4では，提案手法を PC のソフトウェアとして実装する方法について述べ，5では，実装された提案手法を実写映像に適用して評価する．

2 動き領域の分離・追跡手法の概要

本研究では，映像中の動き領域の分離と追跡を以下に示す 1.-5. の処理により実現する．

1. フレーム間の動き推定：フレーム間のアフィン動きパラメータを文献 [1] の手法により推定する．
2. フレーム間での動き物体の分離：処理 1. で得られる動きに従わない画素を取り出して領域併合する．
3. フレーム間の分離結果の投票：各フレーム間で分離された動き領域の画素位置に投票する．
4. 最終的な動き分離結果の出力：時系列方向に沿って安定して高い投票値を得る領域を信頼できる動き領域として出力する．
5. 動き領域の追跡：処理 4. で得られる各フレームごとの動き領域をその重心位置に基づいて追跡する．

連絡先: 興梠 正克　早稲田大学理工学部 村岡研究室
〒 169-8555 東京都 新宿区 大久保 3–4–1
E-mail:kourogi@muraoka.info.waseda.ac.jp

第13回人工知能学会全国大会
（JSAI'99） pp.105-108 (1999)

3　動き領域の分離と追跡

3.1　フレーム間の動き領域の分離

フレーム間の動き領域の分離を以下の 2 つの手順で実現する．

1. 動き領域の候補画素の選別
2. 候補画素がつくる動き領域の併合

3.1.1　動き領域の候補画素の選別

文献 [1] のフレーム間の動き推定手法により得られたアフィン動きパラメータ $(a_1, a_2, .., a_6)$ を以下の式 (1) に代入して，点 (x, y) の各画素上での予測アフィン動きベクトル (u, v) を求める．

$$(u, v) = (a_1 x + a_2 y + a_3, a_4 x + a_5 y + a_6) \qquad (1)$$

点 (x, y) の画素上で動きベクトル (u, v) がフレーム間の画素単位マッチングに失敗するとき，この画素を動き領域の候補とみなして，画素集合 S_e に登録する．なお，有意な情報を得るため，輝度勾配の絶対値が十分に大きな画素だけを対象とする．

画素単位マッチングは，画素の輝度勾配の絶対値が大きいほど動きベクトルの誤差の影響を受けやすい．そこで，勾配による影響を正規化するため，以下の式 (2) が定義する正規化誤差 e を導入する．

$$e = \frac{|I_c(x+u, y+v) - I_r(x, y)|}{|\frac{\partial I}{\partial x}| + |\frac{\partial I}{\partial y}|} \qquad (2)$$

この正規化誤差 e がしきい値 T_e を越えるとき，画素単位マッチングに失敗したとみなす．なお，$I_r(x, y), I_c(x, y)$ はそれぞれ参照フレーム画像と現フレーム画像の点 (x, y) 上の輝度値である．

3.1.2　候補画素の併合による動き領域の分離

3.1.1で述べた手順により得られる動き領域の候補画素の集合 S_e の各画素について，点 $(x, y) \in S_e$ からの距離が T_d 以下にある画素を動き領域の候補領域 R とする．すなわち $R = \{(p, q) | d((p, q), (x, y)) < T_d\}$ である．なお，$d(P, Q)$ は 2 点 P, Q 間の距離関数である．ここでは処理の簡便性から $P_i = (x_i, y_i), P_j = (x_j, y_j)$ のとき，距離関数を $d(P_i, P_j) = max(|x_i - x_j|, |y_i - y_j|)$ と定義する．点 P から距離 T_d 以内の領域は P を中心として一辺が $2T_d$ の正方形領域となる．

2 点 $P_i, P_j \in S_e$ の距離 $d(P_i, P_j)$ が一定のしきい値 T_d 以下であるとき，これらの画素がつくる候補領域 R_i と R_j を図 1に示すように併合する．

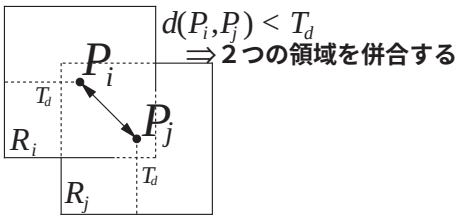

図 1: 動き領域の併合条件

以上に述べた候補領域の併合処理を行うと，いくつかのお互いに共有部分のない領域が得られる．これらのうち，領域内に含まれている点 $(x, y) \in S_e$ の画素数が一定のしきい値 T_p を越える候補領域の併合結果を動き領域として取り出す．

3.2　時系列方向への動き領域の追跡

3.1で述べた動き領域を分離する手法では，フレーム間の動き推定結果の誤差やノイズ等の影響により，動き物体以外の領域を誤って分離する可能性がある．そこで，以下に示す手順 1.-6. によりフレーム間の動き領域の分離結果を時系列方向に沿って追跡する．

1. フレーム間で分離された動き領域の画素位置に投票する．
2. フレーム間で分離された領域上での動きパラメータを推定する．
3. 手順 2. で信頼できる推定結果が得られた場合，領域上の投票値をその動き分だけ移動させる．得られない場合，フレーム間のグローバルな動きパラメータ分だけ移動させる．
4. 蓄積された投票値が十分に大きな画素を取り出して，各画素がつくる領域を併合して動き領域を分離する．
5. 手順 4. により各フレームで分離される動き領域を，その重心位置に基づいて追跡する．
6. 蓄積された投票値を減衰させる．

まず，フレーム画像と同じ $x - y$ 座標系 を持つ投票空間 $V(x, y)$ を定義する．手順 1. では各フレーム間で分離された動き領域上の画素位置 (x, y) に投票する．時系列方向に沿って各フレーム間の投票で蓄積された投票値 $V(x, y)$ を点 (x, y) の動き領域の確信度として利用する．次に，手順 2. では手順 1. で分離された各領域について，その動きパラメータを推定する．手順 3. では，手順 2. の動き推定結果がフレーム間の領域上で良好なマッチング結果を与える場合，その領域の投票値を動き推定結果

に合わせて移動させ，そうでない場合，フレーム間のグローバルな動きに合わせて移動させる．手順 4. では，投票の蓄積値 $V(x,y)$ がしきい値 T_{vote} を越える点 (x,y) を取り出して，3.1.2で述べた方法に基づいて領域を併合する．手順 5. では，手順 4. で分離された動き領域について，各領域の重心位置を求めて，前回のフレーム間で分離された動き領域との重心位置との対応づけにより動き物体を追跡する．最後に，手順 6. では全体の投票累積値を定数 c 倍 $(0 < c < 1)$ して減衰させる．すなわち，以下の式 (3) に基づいて投票蓄積値を更新する．

$$V_{new}(x,y) = c \cdot V_{old}(x,y) \tag{3}$$

ここで，$V_{new}(x,y), V_{old}(x,y)$ はそれぞれ点 (x,y) 上の更新後と更新前の投票蓄積値である．この更新処理で，短い期間に誤って分離された領域の投票結果を取り除くことができる．

点 (x,y) において蓄積された投票値 $V(x,y)$ がしきい値 T_{vote} を越えるとき，その画素を動き領域とみなす．

3.3 処理の計算コスト

前述の動き領域分離の手法は次の 5 つの演算処理から成る．(a) 画素ごとに式 (2) をテストして動き領域の候補となる画素を選び出す．(b) 候補画素の領域を併合して，動き領域を分離する．(c) 分離された動き領域上の画素に投票する．(d) 動き領域上の動きパラメータを推定する．(e) 推定された動きパラメータ分だけ投票値を移動させる．

動き分離で対象とする画素数の上限を N 個，一つの動き領域に併合される可能性がある画素の個数の上限を p 個，フレーム間で分離される動き領域の個数の上限を r 個とすると，本手法が必要とする演算処理とその実行回数の上限は以下の表 1に示す通りである．

演算処理	実行回数 [回]
画素単位マッチング	N
動き領域への投票	N
動き領域の併合	$N \times p$
領域ごとの動き推定	r
投票値の移動	N

表 1: 動き領域の分離・追跡の計算コスト

4 提案手法の PC のソフトウェアへの実装

4.1 実装する計算機環境

提案手法を，汎用的な PC (OS: Linux-2.2.5 SMP 対応, CPU:PentiumII-450MHz [CPUx2], RAM: 256MB) 上でソフトウェア実装した．画像取り込みハードウェアには，BT848 チップを搭載した画像キャプチャカード (Video for Linux 対応) を用いて．画像表示には X11 ライブラリ関数を用いた．取り込まれるフレーム画像は 320×240 24 ビット RGB カラーである．

4.2 実現方法

提案手法を以下の 4 つの処理に分けて，それぞれを独立した UNIX プロセスとして実装し，プロセス間の通信には共有メモリを用いる．

1. フレーム間の動き推定
2. 動き領域の分離
3. 動き領域の投票と追跡
4. 分離・追跡結果の表示

プロセスで実装する利点は，マルチプロセッサ PC 上で実行する場合，OS が対応していれば，プロセッサの台数効果による高速化を期待できることである．

4.3 処理の高速化のための工夫

4.2に示した 4 つのプロセスの流れ処理を単純に実装すると，最も処理速度の低いプロセスに全体のスループットが律速される．実際に，前述の計算機環境での実装では，動き領域分離のプロセス 2. は，動き推定のプロセス 1. の約 2 倍の処理時間を要する．このため，動き推定のプロセスは本来処理可能なフレーム枚数の 1/2 しか処理できない．

そこで，次のような改良を加える．動き推定のプロセス 1. が終了した時点で，動き領域分離のプロセス 2. がまだ終了していない場合，新しいフレーム画像を取り込んで次のフレーム間の動き推定を実行する．プロセス 2. が終了していたら，今までに推定した動きパラメータの累積結果と最初と最後のフレーム画像をプロセス 2. に渡して，動き領域を分離する．このようにして動きパラメータ推定の処理速度を低下させずに全体の流れ処理を実現できる．

5　実験

4で述べた実装において，実写映像を入力として動き物体の分離・追跡をオンライン処理する実験を行い，処理の有効性を示し，その実時間性について検証する．

5.1　動き物体の分離・追跡の結果

動き物体をカメラでフリーハンド撮影した，手ぶれや回転，ズームを含む映像 (人物が両手に持った物体を動かしながら提示している) を入力として，提案手法によりその動き物体を分離・追跡した．入力映像のいくつかのフレーム画像と動き物体の分離・追跡結果を図 2に示す．上段には入力映像に動き物体の追跡結果を枠で囲んだ画像を示し，下段には時系列方向に沿った動き物体の分離結果を示している．ここで，各フレームごとに分離された領域は時系列方向に沿って対応づけされている．この結果より，カメラの手ぶれやズーム等の影響を受けずに，安定して動き物体を分離・追跡できることが確認できる．

図 2: 動き領域の分離・追跡の結果

なお，実験条件としては，動き領域の候補画素の選別では水平と垂直方向に 4 ピクセルごとにサンプルして，領域併合のしきい値を $T_d = 8$，動き領域の抽出時のしきい値 $T_p = 20$，動き領域への 1 回分の投票値を 10 として，安定した動き領域とみなすしきい値 $T_{vote} = 50$，蓄積された投票値を減衰させる定数 $c = 0.95$ とした．

5.2　提案手法の処理時間

提案手法を構成する各プロセスが要した処理時間と全体の処理のスループットを表 2に示した．

この結果より毎秒 10-12 フレームの処理速度を達成できることが確認できる．また，全体の流れ処理のスループットがプロセス全体に要する処理時間の半分になり，

プロセス	処理時間
フレーム間の動き推定	30 msec
動き領域候補の併合	40 msec
動き領域の投票と移動	70 msec
分離・追跡の結果表示	50 msec
合計	190 msec
スループット	70-90 msec

表 2: 本手法の処理時間とスループット

マルチプロセッサ (2 CPU) による台数効果が得られていることが分かる．

6　まとめ

本論文では，フリーハンド撮影されたカメラ映像を入力として，映像中の動き物体を分離して時系列方向に追跡する手法を提案した．提案手法を PC のソフトウェアとして実装して，実写映像をオンライン処理する評価実験を行い，処理の有効性と実時間性を検証した．

参考文献

[1] 興梠, 村岡, “グローバルなアフィン動きパラメータの実時間推定手法”, 信学論 (D-II), vol.J82-D-II, No.7, 1999. (掲載予定)

[2] M. Kourogi, T. Kurata, J. Hoshino, Y. Muraoka, “Real-time image mosaicing from a video sequence”, Proc. of ICIP’99, 1999. (To appear)

[3] M. Irani, B. Roussa, S. Peleg, “Detecting and tracking multiple moving objects using temporal integration”, Proc. of ECCV’92, pp. 282-287, 1992.

[4] J. Y. Wang and E. H. Adelson, “Representing moving image with layers”, IEEE Trans. on Image Processing, 3, 5, pp.625-638, 1994.

[5] 栄藤, 白井, “色, 位置, 輝度勾配に基づく領域分割による 2 次元動き推定”, 信学会 (D-II), vol.J76-D-II, No.11, pp.2324-2332, 1993.