Touch'n Move: 直感的なモバイル映像インタフェース

宮下 広夢[1,a)]　佐藤 隆[1]　東野 豪[1]

概要：映像の時間 (フレーム位置) と空間 (画面上の座標) を対応付けることで，あたかも被写体をつかんで動かしているかのように映像を直観的に再生できる技術がある．しかし従来技術では，コンテンツの作成にかかる編集や計算のコストが高く，撮影してすぐにコンテンツを再生できないといった問題があった．そこで本研究では，事前の動き検出を簡略化し，ユーザのなぞった位置と方向に基づいてフレームの検索と表示を繰り返す Touch'n Move を提案する．これにより，モバイル端末でもコンテンツをすばやく提示することでき，実技をその場で撮影し，直感的に動かして振り返るといったことが可能になる．性能評価では動き情報の検出とユーザの入力による検索のどちらにおいても，高速に実施されることが確認された．

Touch'n Move: A Mobile Video Interface by Direct Manipulation

Hiromu Miyashita[1,a)]　Takashi Satou[1]　Suguru Higashino[1]

Abstract: Users can manipulate a video intuitively as if they were grabbing the objects directly using the ordinary video interface that takes a spatiotemporal slider structures. Creating the slider shape manually is, however, difficult on tablets due to the low resolution of the touch panel. Ordinary methods to generate the slider automatically by analyzing the object motions require much computational time. In this paper, we propose Touch'n Move that reduces preprocessing cost, so that users can manipulate the video directly just after they take it on the tablets. The experimental results revealed the proposed method is fast enough on common tablet.

1. はじめに

映像には人物の運動やモノの移動といった時系列の情報が含まれており，それら動きの情報を利用することで内容に基づいた映像ハンドリングが可能となる．我々は以前からユーザが映像中の動きを簡単に指定することで所望のフレームを表示する映像インタフェースの研究開発に取り組んでいる．その一例として，我々はフレーム画像上でマークするだけで映像中の似た動きを検出するインデクシング手法を提案している [1].

実技や観察など特に動きが重要視される用途においては，映像の動きのある場所や区間を直感的に検索し，再生したいというニーズがある．「画面上の“ ここ ”で動きが発生したとき」というように直感的に指定できれば，該当の動きが発生するまでシークバーを操作してチェックするといった手間を省くことができる．またユーザの入力したポインタに吸い付くように被写体を表示しフレームを遷移させることで，コマ送りや逆再生などをインタラクティブに実現できる．

そこで本稿では，映像に含まれる動きを高速に検出し，ユーザが被写体をつかんで動かすかのように直感的に再生できる Touch'n Move を提案する．Touch'n Move は前処理を簡略化することですばやく直感的な映像コンテンツを提供できることを特徴とし，タブレット端末のみで利用できるためより多様な使い方が期待される．

以下，2 章では関連する技術を紹介する．3 章では従来の課題を解決するアイディアと，具体的な動作フローについて説明する．4 章ではモバイル端末での実装について述べ，5 章では現段階での性能評価について述べる．最後に 6 章ではまとめと今後の展望を述べる．

[1] NTT サービスエボリューション研究所
NTT Service Evolution Laboratories, Yokosuka, Kanagawa 239–0847, Japan
[a)] miyashita.hiromu@lab.ntt.co.jp

2. 従来技術

2.1 直感的な操作を可能にするビデオインタフェース

映像を分かりやすく提示することを目的として，その映像が持つ時間的・空間的な情報を抽出し可視化する技術は以前から提案されてきた [2]．その中に，あたかも被写体をつかんで動かしているかのように映像を直観的に再生できる CyberCoaster がある [3], [4]．CyberCoaster は，被写体の動きの軌跡を模した折れ線型のスライダをフレーム画像上に配置し，時間 (フレーム位置) と空間 (画面上の座標) を対応付けたものである．この直感的な映像操作は実技や観察を伴う授業で特に効果を発揮すると考えられ，小学校の授業に試験的に導入した結果，体育のマット運動技能が有意に向上するなど教育への有効性が認められている [5]．

Dragicevic らは被写体の移動軌跡を自動的に推定してスライダとフレーム画像を時空間的に対応付ける DimP を提案している [6]．Dimp は SIFT による特徴点の検出と特徴量のマッチングを利用しており，各特徴点の移動軌跡をスライダとして多数記憶しておく．ユーザはマウスのクリックで特徴点を指定し，その経路に沿ってドラッグすることで映像をシークできる．

Karrer らが提案した DRAGON では，映像のフレーム画像からオプティカルフローを検出し，折れ線型のスライダを自動的に生成している [7]．DRAGON では解析サーバで計算した結果をモバイルデバイスに送信しタッチスクリーンで動かせる派生型や，ユーザの選択した被写体が画面上で停止している区間もシークできるようにスライダを変形させるタイプも開発されている [8], [9]．

Goldman らは Particle Tracking を利用した画像上の被写体領域の追跡手法を提案し，その応用として被写体へのアノテーションと直接操作による映像のシークを実施している [10]．この手法では表情の変化といった比較的小さい動きの操作や，被写体を本来の位置からずらして再配置するといったことが可能である．

2.2 課題

これらの従来技術には共通して，コンテンツの作成にかかる編集や計算のコストが高く，撮影してすぐにコンテンツを再生できないといった問題がある．

まず CyberCoaster では動きが含まれていない区間など，直感的な映像操作に不要な区間を手動でトリミングしなければならない．さらに折れ線型のスライダの設定では手動でスライダのプロットや時間軸との対応付けをする必要があり，細かい操作が要求される．そのため，指先で操作する座標精度の低いタッチパネルでの編集は困難である．

自動的に被写体の動きを検出する手法では，特徴点の検出や追跡にかかる計算量が非常に多い．例えば 1 フレームあたりの計算時間を参照すると，Dimp では一般的なデスクトップ環境で 5 秒，DRAGON では 37.5 秒，Goldman の手法では 5 分となっている．またこれらの手法ではマウスでクリックした被写体の移動軌跡をスライダとして映像シークに利用している．そのため，クリックする時点でフレームアウトしている被写体は指定できず，一度通常のシークバーや他の被写体を操作して，所望の被写体がフレームインするまでシークしなければならない．

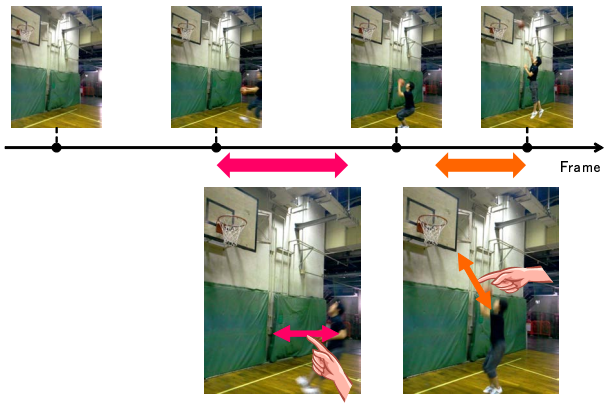

図 1 フレーム画像上で動きを指定することで映像をシークする
Fig. 1 Motion searching by direct manipulation

コンテンツの幅を広げるためには人やモノの移動・運動が活発になる屋外での利用が重要だと考えられ，そのためモバイル端末で撮影から再生まで実現できることが求められる．しかしネットワーク上のサーバに常に接続できるとは限らず，モバイル端末上でのコンテンツ編集や計算量の多い処理は手間がかかるため，使いづらくなってしまう．

3. Touch'n Move の提案

3.1 動き軌跡の予測

我々は CyberCoaster を利用するユーザの多くが，被写体がどのような軌跡を描いて移動・運動するかを映像のフレーム画像から予想し，入力を開始していることに注目した．例えば車は道路に沿って移動する，走者はゴールに向かって走るなど，ユーザは背景や被写体の状況を経験と照らし合わせて被写体の動きを判断できる．予測したような動きが映像に含まれていなかった場合や，フレーム画面上に配置されたスライダの位置が直感とずれていた場合，ユーザは自らなぞる軌跡を修正していた．

一方でうまく被写体をつかんでいるかのようにコンテンツを再生できたとき，ユーザはその軌跡上を何度も往復してなぞることが，今までの実験や利用調査から判明した．特に児童は被写体をつかんで動かしている感覚を面白がり，こするような入力を続けることが多々あった．

そこで Touch'n Move では，従来のように映像中の動きを細かく追跡して事前にスライダを作成するのではなく，ユーザの入力に基づいた検索と表示を繰り返す仕組みを提

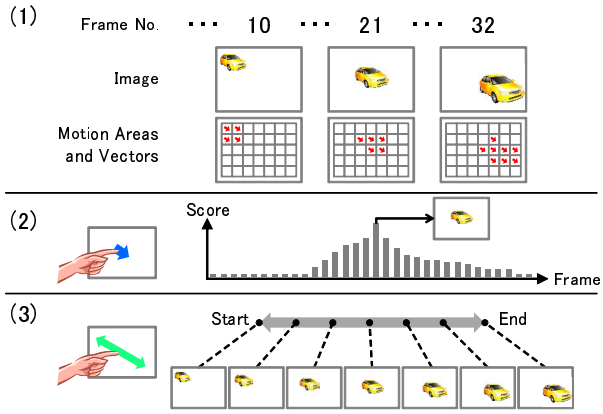

図 2 動きの検出と表示するフレームの決定
Fig. 2 Calculation of motions and detection of display frames

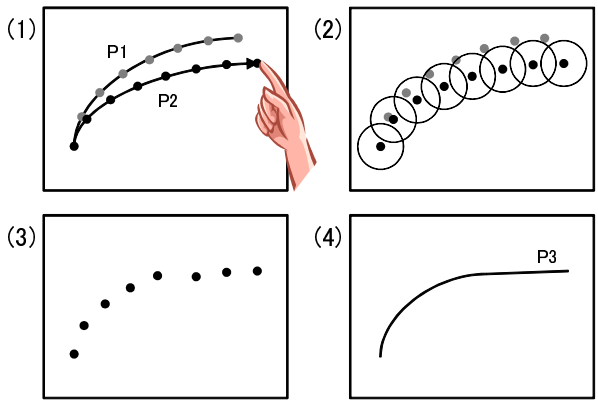

図 3 なぞり入力のマージ
Fig. 3 Paths adjustment

案する．これにより，直感的な映像操作を損なわず，あらかじめ検出する動き情報を簡略化して高速にコンテンツを提供することができる．また表示されているフレームに関わらず映像全体から動きを検索することで，シークをジャンプすることが可能になる．これによりその時点でフレームアウトしていた被写体は操作できなかったという課題を解決する．

図 1 はバスケットボールのシュート動作の映像について，直感的な映像シーン検索を行った例を示している．選手は映像の始めには映っておらず，途中からボールを持ってフレームの右側から現れ，その後にシュート動作を行っている．従来の被写体を追跡してスライダを生成する手法では，選手が登場するまでシークしてからでないと，直感的な操作を開始できなかった．Touch'n Move では，例えばユーザは画面右下を水平方向に，画面中央を上下方向になぞるだけで，それぞれゴール下への踏み込み動作やシュート動作をすぐさま確認できる．

3.2 動き情報とユーザの入力のマッチング

まず映像の各フレーム画像からオプティカルフローを計算し，被写体の画像上での動きを検出する．この計算コストを下げるため，図 2 の (1) のようにオプティカルフローを画像上の一定間隔の座標ごとに求めることで簡略化する．

なお Touch'n Move ではモバイル端末での撮影を前提とするため，検出したオプティカルフローには手ブレに起因するベクトルが含まれている可能性が高い．そこで手ブレをキャンセルするために，似た大きさと方向を持つベクトルが多くを占めていた場合，そのベクトルでもって該当のフレームに含まれるベクトルを減算する．この方法では背景の面積は被写体の面積よりも大きいと仮定しているし，大きなカメラワークではノイズが大きくなってしまう．そのため，できるだけ端末を動かさず，被写体がフレーム内に収まるよう撮影することを前提とする．

オプティカルフローの計算が終わるとディスプレイにフレーム画像が表示され，ユーザは被写体の移動経路を予想してなぞり (スワイプ) 入力を行う．図 2 の (2) のように入力があったとき，その座標における各フレームの動き情報を参照する．その動きのベクトルとユーザがなぞった方向について，似た方向かつより動きが大きい場合にスコアが高くなるよう計算し，スコアが最大となるフレームの画像を表示する．また同時に動きベクトルとなぞった方向が真逆でもスコアが高くなるよう計算することで，時間的に逆方向への運動も検索することが可能になる．これにより，往復したなぞり操作でもタッチしている領域において被写体が動いているフレームが検索・表示され，被写体をつかんで動かしているかのような感覚を得ることができる．

3.3 直感的に動かせる軌跡の推定

ユーザの入力がある度にフレームを検索し表示をするため，動きの情報にノイズや不足がある場合，一方向になぞっているのにフレームが前後することがある．そこで Touch'n Move ではユーザが何度もなぞった軌跡はうまくコンテンツを再生できているパスと判断し，その区間においてフレームを滑らかに遷移させる．

図 2 の (3) のように，何度か往復してなぞり入力が加えられると，その軌跡の始点と終点の座標を取得し，それぞれスコアが最大となるフレーム番号を求める．そしてユーザがその中間をタッチしたとき，始終点との距離の比と対応する中間のフレーム番号を求め，表示する．厳密には被写体が等速運動でない場合はタッチ点と被写体の位置がずれてしまうが，操作性を損なわないと仮定している．

また被写体をうまく動かせるよう，入力を加えるごとに軌跡を微調整する仕組みを導入している．図 3 に示すように，まずユーザの入力をパス (P1, P2) として取得し，各パス上に一定間隔で節を設定する (1)．P2 の各節から近い位置にある P1 の節を参照し (2)，それらの座標の平均値を求め (3)，最終的なパス P3 を出力する (4)．これを繰り返すことでユーザは軌跡を徐々に修正することができる．

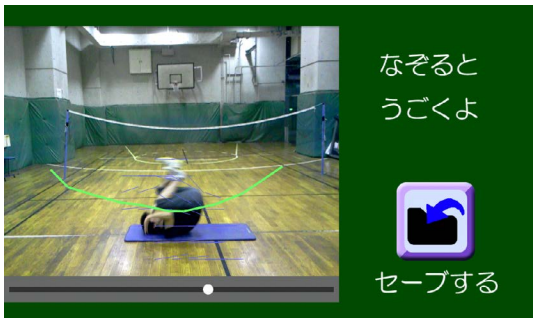

図 4 Touch'n Move アプリケーション
Fig. 4 An example of the application screen

4. 実装

本研究はタッチパネルやカメラを備えるタブレット端末をターゲットとし，Android アプリケーションとして開発している．屋外での利用を踏まえ，ネットワーク上のサーバでの解析はせず，撮影から動きの検出，コンテンツの再生までタブレット端末で実現した．

また画像処理のライブラリとして OpenCV for Android 2.4.2 を利用した．特にオプティカルフローの計算として Pyramid Lucas-Kanade 法を利用した [11]．画像上のオプティカルフローは計算量を増やすほど，動きの検出の精度を高めることができる．本稿ではコンテンツをすばやく提供でき，なおかつ操作性を損なわない程度として，横 19 点，縦 14 点の計 266 点のオプティカルフローを全フレームについて求めるように設定した．

アプリケーションとしての UI や操作方法については，体育など小〜中学校の授業で活用できるよう，児童にも使いやすい構成を心掛けた (図 4)．さらにファイルの送信を提供する暗黙的インテントを組み込むことで，気に入ったコンテンツを友人と共有できる機能を加えた．このときファイルの受け渡しを簡単にするため，撮影した画像と検出した動き情報を ZIP ファイルにまとめている．

5. 性能評価

本研究はコンテンツをすばやくユーザに提供することを目標としている．そこで本稿では動き検出にかかる時間を測定した．具体的には，解像度が 800x600 のフレーム画像から 266 点の動き情報を検出するまでにかかる時間を測定した．その結果，フレーム 1 枚あたり GALAXY Tab SC-01C では 0.34 秒，GALAXY Tab 10.1 LTE SC-01D では 0.11 秒であった．CyberCoaster ではコンテンツの編集に通常 1 分程度の時間を要するが，本手法であれば 100 フレームの映像でも 30 秒強の時間で操作を開始できる．

またユーザの入力と動き情報を参照したスコア計算にかかる時間を測定した，その結果，100 フレーム分に対応する動き情報を対象とした場合でも，GALAXY Tab SC-01C，GALAXY Tab 10.1 LTE SC-01D ともに 10 ミリ秒以下で表示するフレーム番号を求めることができた．これはユーザが入力を加えてからレスポンスを得るまでには十分に短い時間であり，なぞり入力を連続して行っても操作性や使用感を損なわないといえる．

6. おわりに

我々は直感的な映像再生において被写体をつかんで動かしているかのような感覚を損なわず，高速にコンテンツを提供できる技術として Touch'n Move を提案した．本技術ではユーザがなぞりながら被写体の動きを予測・修正することに着目し，入力の位置と方向に基づいてフレームの検索と表示を繰り返すことを提案している．これにより事前の動き検出を簡略化することができ，所望の動きを含む映像シーンをすばやく確認できる．

今後は直感的な映像操作による内容把握への有効性や，学校教育などに適応した場合の使用性などを評価したいと考えている．

参考文献

[1] 宮下広夢, 嶌田聡, 東野豪. ユーザによる動き方と位置の指定に基づいた技能映像インデクシング手法. インタラクション'12, pp. 893–898. 情報処理学会, 2012.
[2] A. Akutsu, Y. Tonomura, and H. Hamada. Video interface for spatiotemporal interactions based on multidimensional video computing. Vol. 1 of *ICASSP-97*, pp. 191–194, 1997.
[3] 佐藤隆, 阿久津明人, 外村佳伸. Coaster：時空間メディアのダイレクトマニピュレーション. インタラクション'99, pp. 65–66. 情報処理学会, 1999.
[4] Takashi Satou, Haruhiko Kojima, Akihito Akutsu, and Yoshinobu Tonomura. Cybercoaster: Polygonal line shaped slider interface to spatio-temporal media. ACM MULTIMEDIA '99, p. 202, 1999.
[5] 宮地功（編）. e ラーニングからブレンディッドラーニングへ. 共立出版, 2009.
[6] Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowitcz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. Video browsing by direct manipulation. CHI '08, pp. 237–246, 2008.
[7] Thorsten Karrer, Malte Weiss, Eric Lee, and Jan Borchers. Dragon: a direct manipulation interface for frame-accurate in-scene video navigation. CHI '08, pp. 247–250, 2008.
[8] Thorsten Karrer, Moritz Wittenhagen, and Jan Borchers. Pocketdragon: a direct manipulation video navigation interface for mobile devices. MobileHCI '09, pp. 47:1–47:3, 2009.
[9] Thorsten Karrer, Moritz Wittenhagen, and Jan Borchers. Draglocks: handling temporal ambiguities in direct manipulation video navigation. CHI '12, pp. 623–626, 2012.
[10] Dan B. Goldman, Chris Gonterman, Brian Curless, David Salesin, and Steven M. Seitz. Video object annotation, navigation, and composition. UIST '08, pp. 3–12, 2008.
[11] Jean yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.