4T-4

ブルーバックを用いないリアルタイム動画編集ソフトの開発

大曽根　直生†　　岡　良祐††　　宮岡　伸一郎†

†東京工科大学 メディア学部メディア学科　　††東京工科大学大学院 バイオ・情報メディア研究科

1. はじめに

現在では映像のデジタル化が進み、個人でも動画の加工・編集を行う機会が多くなってきた。編集した動画はPCに保存したり、動画共有サイトにアップすることで、多くの人を楽しませることができるようになった。しかし、動画編集ソフトを見ると、高品質な動画合成を行う場合は市販のソフトを購入する必要があり、より高品質な合成を行いたければブルーバックを用いたクロマキー合成を行う必要がある。これにより、一般ユーザーが動画合成を行うための敷居が高くなっている。

現在、画像合成ではPoisson画像合成[1]と呼ばれる手法を用いることにより、対象物を包含する合成領域を指定するだけでシームレスに合成することができる(図1)。しかしPoisson画像合成は処理に時間を要するため動画編集に適用するのは困難である。そこで、Poisson方程式を解いたときと同等の解を得ることができるMean-Value Coordinates[2][3](以下MVC)と呼ばれる高速な手法を用いることで、一般ユーザーが用意することが困難なブルーバックを用いなくてもシームレスな動画合成を行うことができる動画編集ソフトの開発を目的とする。

(a)領域指定　(b)単純合成　(c)Poisson画像合成

図1. 合成イメージ

2. 処理の手順

本研究の処理の手順を図2に示す。

まず、ユーザーが合成元動画と合成先動画の2つの動画ファイルを読み込み、合成元動画内で合成したい領域をペンツールを使い指定する。次にMVC合成に必要な重みを計算し、この重みをもとにMVC合成処理をしていく。また、本研究では動画を対象にしているため、Object Tracking処理を間に入れることで、合成対象オブジェクトを自動追跡し、ユーザーの負担を軽減させる。

Compilation without BlueBack
Naoki OSONE, Ryousuke OKA, Shinichiro MIYAOKA
School of Media Science, Tokyo University of Technology, 1404-1 Katakura-machi, Hachioji-shi, Tokyo 192-0982 Japan

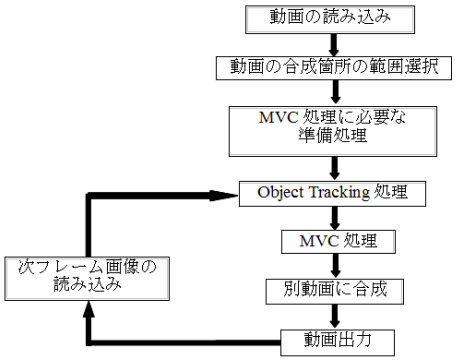


図2. 処理の流れ

3. 合成手法

3.1 Mean-Value Coordinate

現在画像合成の手法としてPoisson画像合成がある。これは貼り付け画像、領域内、貼り付け先画像をそれぞれ f_s、P_s、f_t とする時、境界線上 ∂P_s で合成後の輝度値と貼り付け先画像の輝度値が等しく、領域内部 P_s のラプラシアンを貼り付け画像のラプラシアンと一致するように以下のPoisson方程式を解くことで求めることができる。

$$\Delta f = \Delta f_s \text{ over } P_s \tag{1}$$

$$f|_{\partial P_s} = f_t|_{\partial P_s} \tag{2}$$

また、Poisson方程式を解くことはLaplace方程式を解く事と同義であるので、次式が成り立つ。

$$\Delta \tilde{f} = 0 \tag{3}$$

$$\tilde{f}|_{\partial P_s} = f_t - f_s \tag{4}$$

式(2)、式(4)より、合成画像 f は次のように表すことができる。

$$f = f_s + \tilde{f} \tag{5}$$

MVCとは $\tilde{f}$ をPoisson方程式を解くほどの膨大な計算量は必要なく、求めることができる方法である。貼り付け画像 f_t の内部画素 x において、境界座標を $\partial P_s=(p_0, p_1, \ldots, p_m=p_0)$、$\tilde{f}$ は次のように求められる。

$$\tilde{f}(x) = \sum_{i=0}^{m-1} \lambda_i(x)(f_t - f_s)(p_i) \tag{6}$$

ここで使用する重み λ_i は次の式(7)、(8)によって求めることができる。

$$\lambda_i(x)=\frac{W_i}{\sum_{j=0}^{m-1} W_j} \quad , i=0,1,\ldots,m-1 \qquad (7)$$

$$W_i=\frac{\tan(\frac{\alpha_{i-1}}{2})+\tan(\frac{\alpha_i}{2})}{\|p_i-x\|} \qquad (8)$$

式(8)において、$\angle p_{i-1}, x, p_i$、$\angle p_i, x, p_{i+1}$ の角度をそれぞれ α_{i-1}、α_i としている(図3)。

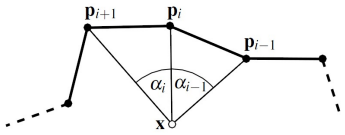
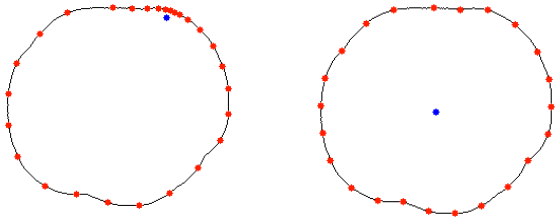


図3. 角度 α_i の定義

3.2 重みサンプリングの最適化

全境界座標を使用し、重み λ_i の計算、合成を行うとPoisson画像合成以上に時間がかかってしまう。そこで、合成領域の内部画素 x と境界点 p_i までの距離を元に λ_i の計算に使用するかの判定を行い、内部画素 x ごとに λ_i の計算に使用する境界点を変える処理を施した。重みサンプリングに使用する点を視覚化した例を図4に示す。青い点が内部画素 x、赤い点が重み計算に使用する境界点 p_i である。



(a)境界点に近い場合　(b)領域の中心付近の場合

図4. 重みサンプリングの例

4. Object Tracking

動画合成において、ユーザーが初期フレームで合成オブジェクトを選択しても、数フレーム経過すると選択したオブジェクトが合成選択領域からはみ出してしまうという問題が発生する。本研究では、Mean-Shift法を用いてユーザーがオブジェクトを選択し直さなくても自動でオブジェクトを追跡し、ユーザーの負担を軽減させる機能を設ける。

5. 実験・評価

Poisson画像合成とMVCの合成の処理時間と生成画像を比較する。1フレームの速度比較をまとめたものを表1に、生成された画像を図5に示す。なお、表中の処理時間についてはミリ秒で表記している。

表1. Poisson画像合成とMVCの速度比較

(a)合成画素7000の時の処理速度

	重み計算	合成時間
Poisson画像合成		65
MVC	295	6

(b)合成画素15000の時の処理速度

	重み計算	合成時間
Poisson画像合成		185
MVC	1000	18

(a)Poisson画像合成の結果画像　(b)MVCの結果画像

図5. 合成結果画像

MVCは合成の前に重み λ_i の計算が必要になるが、合成処理に要する時間がPoisson画像合成の10分の1の時間で合成することができる。また、生成される画像もPoisson画像合成による画像(図5(a))とMVCによる画像(図5(b))でほとんど違いがない。この実験により動画合成に適用できる処理速度であることがわかった。

6. おわりに

本研究では、MVCを用いた動画編集ソフトの機能について提案した。また、物体追跡により、動画合成において発生する合成領域からオブジェクトがはみ出す問題を解決することで、ユーザーの負担を軽減させる機能を設けた。今後はMVCをより高速化する手法を検討し、より大きな動画に対してもリアルタイム性を損なわせることなく合成ができるツールを製作していく予定である。

参考文献

[1] Perez, P., Gangnet, M. and Blake, A.: Poisson Image Editing, Proc. SIGGRAPH'03, pp.313-318,2003

[2] Farbman, Z., et al.: Coordinate for Instant Image Cloning, Proc. SIGGRAPH'09, pp67:1-pp67:9, 2009

[3] Michael S. Floater, :Mean Value Coordinates, Computer Aided Geometric Design, Vol. 20, No. 1, pp19-27, 2003