インターネット動画転送における複数動画の同期転送に関する研究

Study on Synchronous Transmission of Multiple Video in Webcasting

5119E018-9　趙 逸恬　　指導教員　坂井 滋和　教授
ZHAO Yitian　　Prof.　SAKAI Shigekazu

概要：Web 上で複数の動画を同期表示するための最適手法を探し出すために。従来手法の分析によりその問題点について検討を行った上で、WebSocket プロトコルを用いた 3 つの手法を考案した。そして各手法について実験を通じて効果の検証を行い、各手法の利点と欠点および最適手法の選択についての考察を行った。
キーワード：Web コンテンツ配信、WebSocket、画像処理、ビデオ処理、CSS mask-image 属性
Keywords: Webcasting, WebSocket, Image Processing, Video Processing, CSS mask-image Attribute

1. はじめに

インターネットのよる動画配信は、2007 年ニコニコ動画が弾幕を開発して以来、複数の映像同期が行われるようになり、視聴者は意見や感想の共有が実現できるようになった。しかし、弾幕の数が増えるとともに、弾幕が主体を遮ることで画面が見辛くなるという問題が顕在化した。2018 年、Bilibili は「智能弾幕」を開発し、この問題の解決を試みた。しかしこの技術には前景と背景映像がずれる問題点が発生する。本研究は Bilibili の「智能弾幕」におけるこの問題に対して、代用手法を提案し、実験を通じて、最適な手法を探し出すことを目標とした。

2. Web コンテンツにおける映像同期合成

2.1 発展沿革

Web における映像同期は Flash により実現ざれた。しかしその後 HTML5 の登場によりが映像同期は Flash から HTML5 に移行した。 HTML5 は Flash に比べて互換性と性能が優れている。また、Flash における同期は単一であり、それに比べて HTML5 は内蔵 API を複数持つため、多様な同期方法が実現できる[1]。

2.2 先行研究と先行事例

2018 年、Ingar M. Arntzen などの研究者が W3C で「Media Synchronization on the Web」を発表した[2]。彼らは、時間軸を利用し Web コンテンツを同期する手法とそれに対応する API について論じたが、提案手法は全て時間軸に関わっており、その利活用の範囲が限定されている。前に述べた Bilibili の事例では、動画の Video 要素の上にある弾幕 div 要素の mask-image 属性に mask-frame を設定することで、動画の前景と背景を分離する。これによって弾幕過剰問題をほぼ完璧に解決した。しかし Bilibili の手法は前景映像と背景映像がずれる問題点を持っている。その原因は通信プロトコルに依るところが大きい。HTTP プロトコルの通信方式はサーバーとクライアント間の遅延を大きくする原因となる[3]。

3. 提案手法

本研究では通信プロトコルに HTTP に代えて WebSocket を利用する方法を提案する。そしてその中で以下にあげる３手法を提案する。

FV-時間軸同期法は従来手法と異なる点がプロトコルの変更のみである。この手法は HTTP のモラールをまねするため、従来手法の延長線にある改良である。

FF-同期転送法の原理は元動画とそれと対応する mask-frame を同時転送して同期を行う。サーバー側で前処理を行なって元動画の frame と mask-frame を１つの文字列に圧縮する。端末で再生される時にサーバー側に Request を送信し、サーバー側はそれを受けたら Timer を起動し、毎秒 30frame で文字列を端末に送る。端末は文字列を解読し、元動画の frame を div に表示し mask-frame を弾幕 div の mask-image 属性に設定する。

VV-前景背景分離法では、元動画の前景と背景を分離しそれぞれを単独の動画にして同期を行う。前処理を行なって frame 毎に動画の主体部分と他の部分を分離する。その後、主体の部分を透明背景がつく動画に圧縮し、他の部分を主体部分が透明の動画に圧縮する。端末はこの２つの動画を位置とサイズが完全に一致する div に格納し、この２つの動画のメディアコントロールもリンクする。これにより、動画が再生されるに、この２つの動画も同時に再生される。

今回の実験では、上記 3 手法のほかに FF-α チャンネル法を加えた。この方法は、RGB の色彩情報に、α データを加えた 4 チャンネルの画像データを転送し、端末側

で画像合成を行う。前処理として、サーバー側は元動画 frame の RGB チャンネルと mask-frame のデータを数列化し、その MatVector を端末に送る。端末は Opencv.js 画像処理ライブラリが必要となるが、これは事前に配置しておく必要がある。その後、数列データを解読し、その中のチャンネルを元動画 frame に復元し、α チャンネルデータを直接画像にし、mask-frame として使用する。

4. 実験と結果

4.1 実験

本実験における設定条件は、回線速度と CPU 性能の 2 点とし、様々な条件下において HD 解像度(1920x1080)の 8.40 秒(T_0)分（30fps）の動画を使用して、初期化時間（T_i）、動画 1 プレイ完了時間（T_{v1}）、動画 2 プレイ完了時間（T_{v2}）3 つを測定した。評価指標としては、同期率(S)と遅延率(L)を設定した。同期率と遅延率の計算方法を下記の式(1)、(2)に示す。

$$S = \frac{Min(T_{v1}, T_{v2})}{Max(T_{v1}, T_{v2})} \tag{1}$$

$$L = 1 - \frac{T_0}{Max(T_{v1}, T_{v2})} \tag{2}$$

4.2 結果

実験結果を下図に示す。

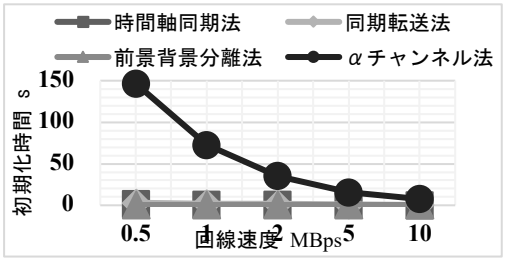

図 1　各手法初期化時間と回線速度の関係

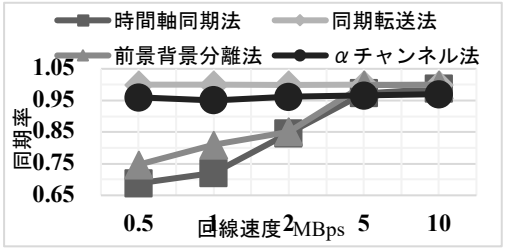

図 2　各手法同期率と回線速度の関係

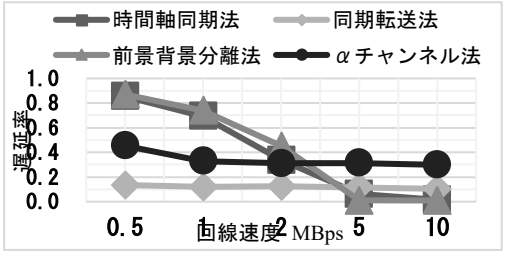

図 3　各手法遅延率と回線速度の関係

5. まとめ

シミュレーションの結果に比べて、実験結果では FF-同期転送法以外の提案手法が完全同期に達することができなかった。その原因は現在本研究が提案した手法の全てにおいて問題点が存在するためである。そして現状を踏まえて最も妥協な手法は FV-時間軸同期法と考えられる。その手法は完全同期ができないがその代わりに実装コストが低く、サービス提供者にとってコストパフォーマンスの良い選択肢である。将来において、各手法の問題が解決され完全同期を実現できる時には、最適方法は 1 つを選択することは難しく、利用目的やインターネットの環境によって相応しい手法を選択することが必要であると考えられる。

参考文献

[1]. W3C, Open Web Platform Milestone Achieved with HTML5 Recommendation, https://www.w3.org/2014/10/html5-rec.html.en

[2]. Ingar M. Arntzen, Njal T. Borch and Franc¸ois Daoust, Media Synchronization on the Web, https://www.w3.org/community/webtiming/files/2018/05/arntzen_mediasync_web_author_edition.pdf

[3]. W3C, HTTP - Hypertext Transfer Protocol, https://www.w3.org/Protocols/

[4]. W3C, The WebSocket API, https://www.w3.org/TR/websockets/