Vol.2020-GN-109 No.26
Vol.2020-CDS-27 No.26
Vol.2020-DCC-24 No.26
2020/1/24

トラスト指向インターネットライブ放送における フレームレート安定化方式

松本哲[†1]　義久智樹[†1]　川上朋也[†2, †1]　寺西裕一[†3, †1]

概要：インターネットライブ配信に写った配信者の顔から視聴者が氏名を特定して脅迫行為を行ったり，配信者の発言を侮辱と捉えた視聴者が周囲の様子から居場所を特定して襲い掛かるなど，配信者に危険が及ぶことが社会問題化している．これらの問題は，配信者と視聴者との間に適切な信頼関係（トラスト）が確保されていなかったことに起因する．著者らはこれまでの研究で，上記の社会問題を解決する為に，配信者の方針に従ってトラストを適切に管理（放棄，構築，維持）するトラスト指向インターネットライブ配信システムの設計を行った．本稿では，トラスト指向インターネットライブ放送において，プライバシー保護の画像処理を行いつつ，フレームレートを安定化させながら放送する方式について述べる．

キーワード：映像配信システム，YouTuber，データ・ストリーミング，ストリーミング配信，ビデオ・オンでマンド

1. はじめに

近年のインターネットライブビデオの普及により，インターネットライブ映像配信に，映像情報を配信したい人々の強い関心が寄せられている．インターネットライブ映像情報を配信する殆どの人は，カメラで自分自身をビデオ撮影する. 例えば, YouTube を利用する映像配信者は YouTuber と呼ばれ，100 万人以上いるとされている．YouTuber の殆どが，自らを撮影したビデオを配布している．映像配信者は，配信者と視聴者の間の信頼関係に配慮しながら，個人情報に関連する地域，顔や現在の場所，公序良俗に反しないように映像情報に配慮して配信する必要がある. そこで，著者らは, [1] で提案した，トラスト指向型のインターネットライブ映像配信により，より安全で広く使用されるインターネットライブ映像配信を目指す.

映像配信者が自身で撮影を行う場合，最もプライバシー保護の必要が高い映像情報は配信者の顔である．公開を避ける為の手段として，顔の画像を別のものと置き換える，マスクをする等の映像効果を付加する事が挙げられる．そのような映像効果には，幾つかの処理が含まれている．例えば，顔を検出する，マスク画像の作成，非表示の描画領域の決定，人の顔にのみマスク画像を被せる等があり，高い計算負荷のかかる処理と比較的計算負荷が低い単純な処理の場合がある．

映像効果の為の映像加工処理の計算時間が，再生時の映像フレーム間隔より長い場合，処理された映像の 1 画面毎に描画するフレームレートに遅延が発生し，フレームレートが低下する．この映像加工処理の時間は，処理の複雑さに依存し，描画処理の遅延により，不安定なフレームレートとなる．不安定なフレームレートの映像は，視聴者をとても煩わせる為，インターネットライブ映像配信には安定したフレームレートでの配信が必要となる．

映像加工処理時間を短縮する為の，様々な手法があり，提案されている．それらの幾つかの手法は，映像加工処理時間に上限を設ける手法である．配信者の顔の非表示や顔を変更する映像処理している場合において，処理時間が上限に達した際に，処理のキャンセルを行う場合，トラスト指向のインターネットライブ映像配信であっても，個人情報を公開してしまい，配信者と受信者の間のトラスト関係は破棄されてしまう．配信者と受信者の間にトラスト関係がなかった場合は，映像内の個人情報は非表示にするべきである．従来の映像加工処理時間に上限を設ける手法のみでは，トラスト指向のインターネット映像配信は実現できない.

そこで，本論文では，トラスト指向型のインターネットライブ映像配信において，フレームレートを安定化させながら配信する映像処理システムを提案する．映像加工の条件は，配信者と受信者の間のトラストの関係の有無と，配信者の於かれている状況下での受信者へのトラストの関係の有無で分けられる．前者の場合，配信者の個人情報は常に非表示にでき，提案するシステムでは，映像加工処理時間がフレームレートを超える場合に，ビデオフレームレート間隔に依存して，その間隔内で映像加工処理ができる単純な処理へと変更する事で，安定したフレームレートでの配信を実現する．

本論文では，次の様な，実装・評価をした．動画像のフレーム描画にて，フレーム間隔の時間が短い場合，映像加工処理計算機が実行する予定の，動画中の配信者の顔を検出する処理をキャンセルする．次に，画像の全領域をぼかす単純な処理を実行するだけの処理に置き換え，実装・評価した.

本論文では，特に次の 3 点について述べる．1）トラスト

†1 大阪大学
Osaka University.
†2 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
†3 情報通信研究機構
National Institute of Information and Communications Technology

ⓒ 2020 Information Processing Society of Japan

Vol.2020-GN-109 No.26
Vol.2020-CDS-27 No.26
Vol.2020-DCC-24 No.26
2020/1/24

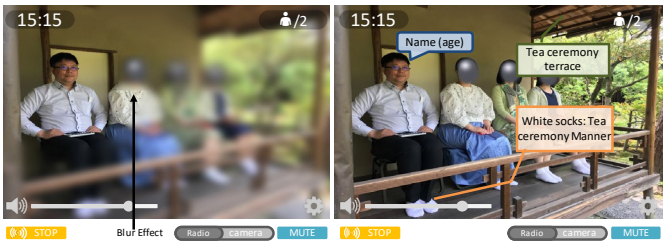

図 1: 非トラスト状態のライブ映像 (左) , トラスト状態 (右).

指向のインターネットライブ映像配信のフレームレートを安定化させる提案，2）提案システムの設計と開発，3）提案システムの評価

本論文は次のように構成される．第 2 章で関連研究について述べる．第 3 章では，トラスト指向インターネットライブ映像配信の為の映像処理システムの提案を述べる．第 4 章で提案システムについて議論し．第 5 章で，システムの評価と実験結果について説明し，第 6 章にてまとめを述べる．

2. 関連研究

負荷分散ストリーム処理（DSP），ピアツーピア（P2P），クラウド， エッジ， またはフォグコンピューティングモデル等[3-13]、多くの負荷分散処理システムが提案されている．幾つかの提案システムでは，オープンソースのストリーム処理に向けて実装されている．Apache Hadoop [14]、Storm [15]、Flink などのプラットフォーム[16], および Spark Streaming [17]， Lopez らによる，ネットワークトラフィックの脅威検出に関する 2 回の実験では，スループット効率と回復力を評価した．ネイティブストリーム処理の性能において，Storm、Flink、および Spark Streaming[18]でのノード障害の評価結果では，Storm と Flink のシステムは、マイクロバッチ処理システム，Spark Streaming，よりも，最大 15 倍高い性能を得ている．一方，Spark Streaming はノード障害に対して強固であり，損失なしで障害復帰を提供する．一部の DSP システムではリアルタイム用に設計されている．ライブ映像ストリーミング処理に適用できる．[5]で，提案された手法では，連続クエリの条件式で，処理時間を短縮できる． その評価より，提案されている手法がホップ数と IoT 環境における，通信トラフィックにおける平均通信量の最大数を減らせることを示している．Ning らが モバイル向けストームとしてリアルタイムストリーム配信処理システムをクラウド利用により提案した[7]． Mobile Storm は計算資源をリモートサーバーにオフロードせずに，ローカルネットワーク内のモバイルデバイスのクラスターを利用する．Mobile Storm 処理の為のアプリケーションは，Android スマートフォンに実装され，また，その評価の為に開発された．評価結果は，リアルタイムに適した解像度でも，様々なフレームレートであっても，ビデオストリーム

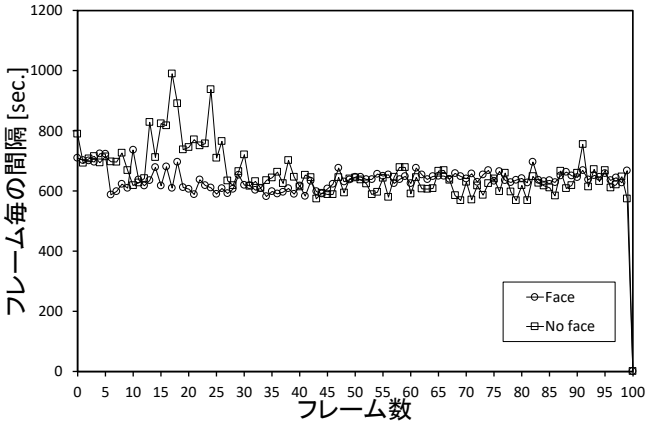

図 2: フレーム間隔の例

を制御できるとの結果を得た．Choi らは，IOT 機器に効果的な，軽量なストリーミング処理のフレームワーク[9] を提案している．DART では, データソースの論理グループ，つまり，モノのクラウド（CoT）が完全に分散された方法で，データストリームをより効率的に処理する為に構成されている． DART は、IoT デバイスを CoT としてグループ化することにより，サーバー利用を基本とした方法とエッジのみを利用した方法の両方を克服することを目的とする．RIDE は，分散環境でリアルタイムの大量の画像ストリームを効率的に処理する為に提案された[10]．RIDE は，アプリケーション，マスター，バッファー，ワーカーの 4 つのレイヤーで構成されている．分散ノード上のタスク間の通信オーバーヘッドを最小限に抑える為に，RIDE では，ワーカーノードにストリームのパーティションを割り当てることにより，粗粒度の並列処理が実現される．さらに，各ワーカーノードでタスクを並列処理することにより，きめ細かい並列処理が実現される．Yang らは，分散フォールトトレラント処理（DFP）メソッドに焦点を当て，スマートコンピューティングのクラウドコンピューティングベースのマルチカメラシステム用に設計された分散画像検索メソッドを提案した[11]. クラウドストレージテクノロジー，データ暗号化，データ取得テクノロジーの組み合わせにより，マルチカメラリソースの効率的な統合と管理が実現する．[12]では，典型的なビデオ分析アプリケーションの処理および帯域幅の問題が調査され，エッジとクラウド間のメソッドの配置決定を理解するのに役立つ． [12]の著者は、プライバシー，集中管理共有，エッジデバイスのメンテナンス，処理および帯域幅のコストなど, [12]で行われたものよりもさらに考慮すべき事項があると述べている．

リアルタイムビデオデータのセキュリティまたはプライバシーに関連して，Liu らは，リアルタイムビデオデータの安全な共有と検索の為のインフラストラクチャを提案した[19]．提案されたインフラストラクチャは，5G テクノロジーとクラウドコンピューティングプラットフォームを展開することから，モバイルユーザーに最適である．デー

ⓒ 2020 Information Processing Society of Japan

Vol.2020-GN-109 No.26
Vol.2020-CDS-27 No.26
Vol.2020-DCC-24 No.26
2020/1/24

タの機密性は暗号化アルゴリズムによって保護されている為，クラウドサーバーがクラッキングされた場合でもセキュリティが保証される．さらに，提案されたインフラストラクチャでは，ユーザー自身のビデオデータ内で安全な検索機能も提供される．Wang らは，OpenFace を提案した．その精度は，入手可能な最高の精度のオリジナルの認証器となっている[20]．OpenFace をフレーム間トラッキングと統合して，RTFace も作成されている．RTFace は，指定されたポリシーに従って全フレームレートで顔を選択的にぼかす，ビデオストリームを変性させるメカニズムである．[20]では，RTFace を使用した大規模カメラネットワーク向けのプライバシー対応アーキテクチャが紹介された．上記の既存の技術らは，計算および通信の負荷を処理ノードに効率的に分散することにより，ストリーム処理時間を短縮できる．ただし，これらの手法は処理時間に上限を設けておらず，フレームレートが変動する可能性が大いにある．本論文は，許容処理時間や適応タスク変更などの状況を考慮した位置づけである．

3. トラスト指向映像配信の提案

本章では，[1]で提案されている信頼指向のインターネットライブビデオ配信システムについて簡単に説明する．

トラスト指向のインターネットライブ映像配信では，配信状況は 2 つの状況に分類される．一方では，配信者と視聴者の間に信頼関係がなく，非トラスト状態と呼ばれる状況である．もう一方は，トラストがある状況であり，トラスト状態と呼ばれる．インターネットのライブ映像配信者が属する状況は, さまざまな要因によって異なる. 例えば，インターネットライブ映像配信の視聴者が配信者の友人だけに制限されている場合，トラストは信頼できる状況である．視聴者が配信者の見知らぬ人である場合，状況は信頼できない状況である．単純なシステムでは，現在の状況が属する非トラスト/トラスト状況は、配信者によって手動で選択される． 視聴者，場所などに関する情報を利用して，自動選択も可能である．図 1 は，信頼できない状況と信頼できる状況でのライブビデオの画像の例を示している．トラスト指向のインターネットライブ映像配信には,「非公開情報」,「極限情報」,「情報暴露」という 3 つのポリシーがある．信頼できない状況では，映像配信者は非公開情報ポリシーを使用できる．このポリシーでは，処理コンピューターが処理を実行して，個人情報閲覧を閉ざします．映像配信会社の個人情報は隠匿されている為，映像配信会社は撮影されたライブ映像を安全に配信できる．ただし，視聴者は映像配信者に関する個人情報を取得できない為，トラストを構築できる可能性は低下する．極限情報ポリシーでは，処理コンピューターが映像配信者によって受け入れられた処理を実行する．映像配信者は，信頼を維持することを目的として，このポリシーを選択する．トラスト状況下では，映像配信者は情報暴露ポリシーを使用できる．このポリシーでは，処理コンピューターが処理を実行して個人情報を公開する．視聴者は映像配信者に関する個人情報を取得できる為, 信頼を構築できる可能性が高まる. ただし，この時のライブ映像配信は安全では無くなる．

トラスト指向のインターネットライブ映像配信では，処理コンピューターは，映像配信者によって選択されたポリシー（非トラストまたはトラスト）に基づいて，映像処理を実行させる．ただし，従来の信頼指向のインターネットライブ映像配信システムでは，処理時間とフレームレートが通常変動することを考慮していなかった．

4. 提案システムについて

本章では，フレームレートを安定化する為に提案されたビデオ処理システムについて説明する．最初に研究目的とその問題を説明し，後に，問題を解決する為のアプローチについて説明する．

4.1 問題点

1 章にて述べた通り，長時間の映像加工処理は不安定なフレームレートを引き起こし, 視聴者を煩わせる. 例えば，図 2 にフレーム間隔と処理の関係を示す．横軸はフレーム番号である．この例では，処理コンピューターが顔を検出し，検出された領域をぼかす為の処理を実行する．「Face」は，ライブ映像に配信者の顔がある場合のフレーム間隔を示す.「No face」は，映像中に配信者の顔が無かった場合を示す．この例に示すように，フレームの間隔は変動し，この場合約 200 [msec.] の範囲となる. この様な間隔の大きな変化は，映像が途切れて見える等，視聴者を煩わす大きな要因と成り得る．そこで，本論文では，安定したフレームレートを目指した映像処理システムを提案する．映像加工処理時間がフレームの間隔よりも短い場合，システムは一定の時間待機することにより，次のフレームの画像を描画する時間を制御できる．でなければ，次のフレームの画像を描画する時間が遅れ, フレームレートが変化する. また，著者らは，処理時間が長い場合でも，フレームレートを安定させる為の映像処理システムを提案する．

4.2 研究の観点

トラスト指向のインターネットライブ映像配信のフレームレートを安定させる為に，2 つのアプローチを採用している．

1 つ目は，映像加工処理時間の短縮である．前節で述べたように，提案システムでは，処理時間がフレームの間隔よりも短い場合，フレームレートを安定させることができる．従って，フレームの間隔よりも短い映像加工処理時間により，安定したフレームレートで配信する事が可能とな

ⓒ 2020 Information Processing Society of Japan

Vol.2020-GN-109 No.26
Vol.2020-CDS-27 No.26
Vol.2020-DCC-24 No.26
2020/1/24

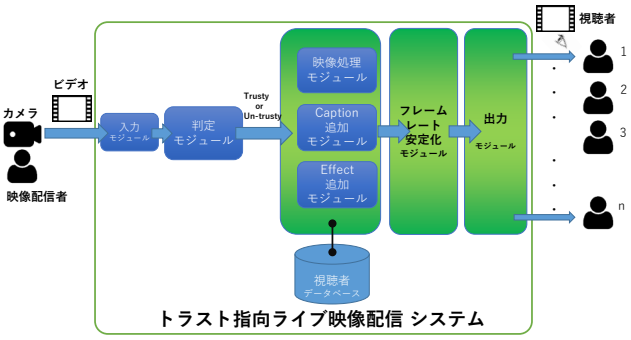

図 3: 提案システムの構成

る．映像処理時間を短縮する為の様々な手法が提案されている．ただし，それらはトラスト指向のインターネットライブ配信に焦点を合わせておらず，システムがこれらの手法を採用している場合でも，幾つかの一般的な懸念が発生する可能性がある．最もプライバシー保護の必要性が高い個人情報は顔である．その為，映像配信者の顔を隠すことに焦点を当て，トラスト指向のインターネットライブ映像配信で顔を隠す為の映像処理時間の短縮方法を提案する．もう 1 つは，映像処理を単純な映像処理に変更することである．1 章で述べたように，ほとんどの映像効果は，単純な処理と比較すると高い計算負荷が必要である．従って，提案手法では，映像加工処理時間がビデオフレームの間隔を超えようとしている時に，処理コンピューターがより短い処理時間で処理を実行できる映像効果の処理に変更する．単純な映像処理の例としては，画像の領域全体をぼかす処理がある．単純な映像処理の処理時間は，複雑な映像処理と比較すると，処理時間が短い為，システムは次のフレームの画像を描画する前に処理を終了し，フレームレートを安定させることができる．

4.3 提案システムの構成

図 3 に，提案システムの構成を示す．提案している構成では，配信者は，トラストを重視したライブ映像配信システムを使用して，撮影ライブ映像を視聴者に配信する．状況判別モジュールでは，視聴者データベース及び，その他の情報を使用して，現在の状況が非トラスト状態であるかトラスト状態であるかを判断する．映像処理モジュールでは，指定された映像加工処理を実行する．視聴者の一部は，ライブビデオを見る前にシステムにログインするので，判別モジュールで活用できる．これらのモジュールの詳細は，[1]に記されている．

提案システムに新たに追加されたのが，フレームレート安定化モジュールである．フレームレート安定化モジュールは，映像処理モジュールを管理し，出力モジュールのフレームレートをチェックする．出力モジュールは，ライブ映像を視聴者に配信するシステムモジュールである．フレームレートが変動する場合，フレームレート安定化モジュールは，映像処理モジュールで実行される映像処理を単純な処理に変更し，処理時間の短縮を試みる．処理された映像データは出力モジュールに転送され，視聴者に配信される．

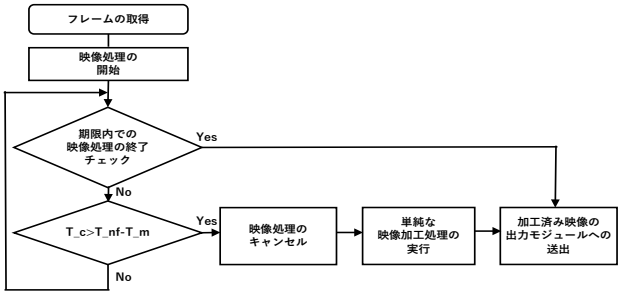

図 4: フレームレート安定化モジュールのフローチャート

4.4 顔映像の高速な隠匿手法

本節では，配信者の顔の非表示を高速に行う提案を行い，その方法について説明する．

4.4.1 顔映像の位置取り

ライブ映像配信者の殆どは自分で自身を撮影する為，顔の位置は殆ど変わらない．例えば，映像配信者はカメラを三脚で部屋に固定し，カメラの正面に座って自分自身を撮影する．この場合，上半身を動かしたときに位置が揺れるが，ビデオ画像での顔の位置はさほど変化しない．別の例として，映像配信者はスマートフォンを使用してライブ映像を配信する場合がある．彼らはスマートフォンを握ってカメラを自分自身に向け，歩きながら自分自身を撮影する．この場合の映像配信者は通常，ビデオ画像の中心に顔がある様にし，そう頻繁には位置を大きく変化させない．顔の位置が大きく変化しない場合，次のフレーム画像の顔の位置は，現在のフレーム画像の顔の位置の近くとなる．これを使用してフレーム内での範囲を縮小して顔を検出すると，画像が小さいほど処理時間が短くなる為，映像処理時間を短縮できる．ただし，縮小画像に顔が含まれない可能性が高くなる為，顔を検出する範囲が狭くなると，顔の検出に失敗する可能性が高くなる．従って，提案する方法では，現在のビデオ画像内の顔の位置からマージンを取り，その範囲内の顔を検出する．

4.4.2 顔映像領域の判別

X_c, Y_c は現在のビデオフレーム画像で検出された顔の領域の左上隅を示し，W_c ，H_cは領域の幅と高さを示めす．領域の左右（M_X）と領域の上下（M_Y）に同じマージンを設定する．次に，次のビデオフレーム画像X_d，Y_d，W_d，H_dで顔を検出する領域は次のようになる．

$$\begin{aligned} X_d &= X_c - M_X \\ Y_d &= Y_c - M_Y \\ W_d &= W_c + 2M_X \\ H_d &= H_c + 2M_Y \end{aligned} \qquad (1)$$

ⓒ 2020 Information Processing Society of Japan

Vol.2020-GN-109 No.26
Vol.2020-CDS-27 No.26
Vol.2020-DCC-24 No.26
2020/1/24

この提案手法では，顔を検出する領域は上記の式で決定される．顔を検出する為の領域を縮小することの有効性を後の評価の節で述べ，この確認をする．

4.5 映像変更処理

本節では，映像加工処理を変更してフレームレートを安定化する為の提案方法を説明する．

4.5.1 映像変更タイミング

提案手法においては，フレームレート安定化モジュールにて，映像加工処理の処理時間がフレームの間隔を超えると，その処理を，映像処理モジュールが，より短い処理時間で処理を実行できる単純な処理に変更する．映像処理を単純な処理に変更した場合でも，映像加工処理時間が発生し，時間はかかる．従って，提案された方法では，単純なビデオ処理にマージン時間を与える． Tm はマージン時間を示す．現在の時刻が次の不等式を満たしている場合，フレームレート安定化モジュールは映像処理を単純なものへと変更する．

$$Tc > Tnf - Tm \quad (2)$$

ここで，Tc は現在の時間，Tnfは次のフレーム画像を描画する時間とする．つまり，$Tnf - Tm$ の値は元の映像処理のデッドラインとなる．提案方法では，マージン時間内に単純な映像処理を完了するのに十分な Tm を決定する．例えば，著者らの試行に於いて，ビデオ処理システムで領域全体をぼかす最大処理時間は 129 [msec.]となった．この施行に於いては，時間は多少変動するが，単純な映像処理を完了するのに十分な時間を予測でき，129 [msec.]より大きい Tm の値で十分となる．

4.5.2 顔映像の隠匿処理

図 4 は，フレームレート安定化モジュールのフローチャートを示している．モジュールがカメラからビデオフレームイメージを取得すると，他のスレッドのイメージに対する映像加工処理を開始する．処理を管理する為に，映像処理は並行して実行され，フレームレート安定化モジュールの他のスレッドで実行される．その後，モジュールは処理が期限内に終了するかどうか，及び，上記の不等式が満たされるかどうかを継続的にチェックする．処理が終了すると，モジュールはビデオフレームの処理済み画像を出力モジュールに転送する．上記の不等式が満たされると，モジュールはその重い映像処理をキャンセルし，単純な映像処理の実行を開始する．単純な映像処理が終了すると，ビデオフレームの加工済みイメージを出力モジュールに転送する．

5. 評価結果について

本節では，提案システムの性能を評価する為に，フレームの間隔と映像加工処理に関する幾つかの評価結果を示す．

5.1 実験環境の設置について

撮影したビデオ画像の顔を検出し，検出された領域をぼかすビデオ処理システムを実装した．映像処理を終了する時間がデッドライン（式（2））よりも長い場合，システムは処理をキャンセルし，画像の全領域をぼかす単純な処理に変更する．このシステムにはライブビデオを配信する機能があるが, この機能はこの研究に直接関係していない為，特筆しない．提案手法の有効性を示す為に，フレームの間隔を示す．この評価では，ビデオのフレームレートを f[fps]に設定した．提案された方法では，現在処理中の処理時間が期限を超えると，処理コンピューターは映像処理を単純な処理に変更する．従って，フレームの間隔が $1/f$ [msec.]より短い場合，提案手法は次のフレーム画像の描画を待つことで一定のフレームレートを実現することができる．この評価では，変化する処理の比率を使用する．この比率とは，映像処理が単純な処理に変更される比率を意味し，単純な映像加工処理の数をすべての映像加工処理の数で割ったものである．値が大きいほど，より多くのビデオフレームが完全にぼやけていることを意味する（領域全体がぼやけています）. 個人情報に関連する地域だけがぼやけている為，トラスト指向のインターネットライブ配信では値が小さいほど良質となる．

5.2 実験環境

評価には，開発したビデオ処理システムを使用する．Visual Studio 2017 を使用してシステムを開発し，システムは Open CV 4.1.0 を使用してカメラから画像を取得し，画像内の顔を検出した．処理コンピューターはラップトップ PC（CPU：Core i7@2.4GHz，メモリー：8GB）である．ラップトップ PC に装備されたカメラを使用して，640x480 RGB（32 ビット）画像を取得する．顔検出には，Open CV に実装されている detect Multi Scale 関数を使用する．

5.3 実験結果の評価

本節では，幾つかの評価結果を示す．はじめに，フレームの間隔を示し，2 つのアプローチの有効性を評価する．その後，処理の領域サイズを変更する性能を示し，領域サイズを変更して顔をより速く隠すアプローチを評価する．次に，顔検出の幾つかのパラメーターを変更した性能を示す．

5.3.1 フレーム間隔について

この研究の目標は，フレームレートの安定化である．これは，フレームの間隔がより一定であることを意味する．従って，フレームの間隔を測定した．図 5 に，フレームの間隔を示す．横軸はフレーム番号，縦軸はフレームの間隔とする．図の「Proposed（f[fps]）」は，フレームレートが f に設定されている場合の，提案された方法でのフレームの間隔を示す． 「Conventional」とは，従来の方法，つまり提案された方法を使用しない場合のフレーム間隔を示す．この方法はフレームレートを考慮しない為，従来の方法で

ⓒ 2020 Information Processing Society of Japan

Vol.2020-GN-109 No.26
Vol.2020-CDS-27 No.26
Vol.2020-DCC-24 No.26
2020/1/24

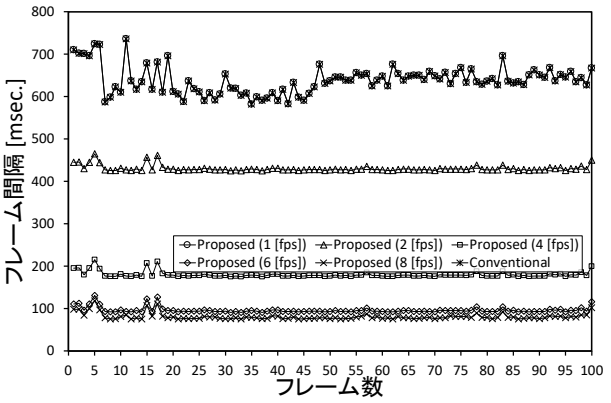

図 5: フレーム数とフレーム間隔

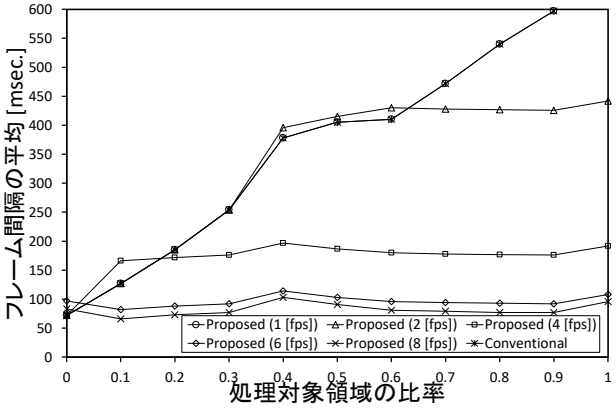

図 6: 対象領域サイズの異なる下での平均フレーム間隔

の結果はフレームレートに依存しない．マージン時間は 130 [msec.]とした．この結果から，提案手法は多くの場合，フレームレートの逆値よりも短い間隔を達成することがわかる．これは，提案された方法が安定したフレームレートを提供することを意味する．ただし，ビデオ処理時間が予測マージン時間よりも長い場合，フレームの間隔が長くなることがある．従来の方法でも，ほぼ 1 [fps]を達成できている．フレームレートが 1 [fps]の場合，映像処理は期限前に終了する為，提案された方法でのフレームの幾つかの間隔は従来の方法での間隔と同じとなっている．ただし，変更前の映像処理が単純な映像処理よりも早く終了する場合，提案手法のフレーム間隔は従来手法よりも長くなる．例えば，1 [fps]の場合，100 フレーム中の 6 フレームのビデオ処理時間は，次のフレームの画像を描画する時間を超える．4 [fps]の場合，100 フレーム中の 19 フレームの映像加工処理時間は，次のフレームの画像を描画する時間を超える．マージン時間を増やすことで，加工処理時間が次のフレームの画像を描画する時間を超える為，フレームの数を減らすことができる．フレームレートを大きくすると期限が早くなる為，処理を変更する割合はフレームレートに比例して増加する．例えば，1 [fps]の場合，10 個の映像処理が 100 フレームで単純な映像処理に変更される．これは，処理を変更する比率が 10/100＝0.1 であることを意味する．4 [fps]の場合，すべての映像加工処理は単純な映像加工処理に変更され，比率は 1.0 となる．従って，提案された方法では，安定したフレームレートを提供するが，処理の変更の割合が増加する．以下の評価結果では，フレームの平均間隔を使用する．

5.3.2 処理対象領域のサイズについて

処理の対象領域サイズを小さくすると，映像処理を実行する為のデータ量が減少する為，ビデオ処理時間が短くなる．従って，処理の対象領域サイズを変更して，フレームの平均間隔と処理を変更する割合を測定した．図 6 は，処理のさまざまなフレームサイズでのフレームの平均間隔を示す．横軸をフレームレートとする．フレームレートは，全フレーム内の処理のフレームレートであり，全フレーム内のピクセル数で処理のフレーム内のピクセル数を割った値である．縦軸は，フレームの平均間隔である．マージン時間は 130 [msec.]である．凡例は以前の結果と同様である．提案手法では，フレームレートが小さい場合，ビデオ処理を実行する為のデータ量が増加する為，フレームレートが増加するにつれてフレームの平均間隔が増加する．フレームレートが大きい場合，映像加工処理時間がデッドラインに達すると映像加工処理が単純な映像加工処理に変更される為，提案された方法ではフレームのほぼ一定の平均間隔を与える．フレームレートが増加すると，ビデオ処理時間が早く期限に達する．従来の方法でのフレームの平均間隔は，この方法では期限が考慮されない為，フレームレートに比例して増加する．例えば，4 [fps]の場合のフレームの平均間隔は約 200 [msec.]となる．これは 250 [msec.]（4 [fps]未満のフレームの間隔）より小さく，提案された方法は安定したフレームレートを与えることができる．一方，従来の方法では，4 [fps]のフレームレートを達成する為に，フレームレートは 0.23 未満でなければならない．

6. まとめ

本論文では，トラスト指向のインターネットライブ映像配信のフレームレートを安定化する映像処理システムを提案する．提案するシステムでは，映像処理の画像領域を変更し，画像加工処理時間がビデオフレームの間隔を超えると，映像処理も変更する．提案されたシステムを実装し，評価した結果から，提案システムがトラスト指向のインターネットライブ映像配信におけるフレームレートを安定化できることが明らかになった．将来的には，映像加工処理時間をさらに短縮し，実際のインターネットライブ映像配信のフレームレートを調査する．

謝辞 本論文の一部は科学研究費補助金（課題番号 JP18K11316），および，I-O DATA 財団の研究助成よる成果である．ここに記して謝意を表す．

ⓒ 2020 Information Processing Society of Japan

Vol.2020-GN-109 No.26
Vol.2020-CDS-27 No.26
Vol.2020-DCC-24 No.26
2020/1/24

参考文献

[1] T. Yoshihisa, S. Matsumoto, T. Kawakami, and Y. Teranishi, Trust-oriented Live Video Distribution Architecture, in Proc. IEEE Int'l Conf. on Computers, Software and Applications, pp. 938-939(2019)

[2] S. Matsumoto, T. Yoshihisa, T. Kawakami, and Y. Teranishi: A Distributed Multi-Viewpoint Internet Live Broadcasting System with Video Effects, in Proc. International Workshop on Informatics (IWIN'18), pp. 83-88 (2018).

[3] X. Zhao, H. Ma, H. Zhang, Y. Tang, and Y. Kou, HVPI: Extending Hadoop to Support Video Analytic Applications, in Proceedings of the 8th IEEE International Conference on Cloud Computing (CLOUD 2015), pp. 789-796 (2015).

[4] W. Kou, H. Li, and K. Zhou, Turning Video Resource Management into Cloud Computing, Future Internet 8(3), 35, 10 pages (2016).

[5] T. Yoshihisa, T. Hara, A Low-Load Stream Processing Scheme for IoT Environments, in Proceedings of the 2016 IEEE International Conference on Big Data (Big Data 2016), pp. 263-272 (2016).

[6] N. Chen, Y. Chen, Y. You, H. Ling, P. Liang, and R. Zimmermann, Dynamic Urban Surveillance Video Stream Processing Using Fog Computing, in Proceedings of the 2016 IEEE Second International Conference on Multimedia Big Data (BigMM 2016), pp. 105-112 (2016).

[7] Q. Ning, C.-A. Chen, R. Stoleru, and C. Chen, Mobile Storm: Distributed Real-Time Stream Processing for Mobile Clouds, in Proceedings of the 4th IEEE International Conference on Cloud Networking (CloudNet 2015) (2015).

[8] T. Li, J. Tang, and J. Xu, A Predictive Scheduling Framework for Fast and Distributed Stream Data Processing, in Proceedings of the 2015 IEEE International Conference on Big Data (Big Data 2015), pp. 333-338 (2015).

[9] J.-H. Choi, J. Park, H. D. Park, and O.-G. Min, DART: Fast and Efficient Distributed Stream Processing Framework for Internet of Things, ETRI Journal, Vol. 39, No. 2, pp. 202-212 (2017).

[10] Y.-K. Kim, Y. Kim, and C.-S. Jeong, RIDE: Real-Time Massive Image Processing Platform on Distributed Environment, EURASIP Journal on Image and Video Processing, 13 pages (2018).

[11] J. Yang, B. Jiang, and H. Song, A Distributed Image-Retrieval Method in Multi-Camera System of Smart City Based on Cloud Computing, Future Generation Computer Systems, Vol. 81, pp. 244-251 (2018).

[12] L. O'Gorman and X. Wang, Balancing Video Analytics Processing and Bandwidth for Edge-Cloud Networks, in Proceedings of the 24th International Conference on Pattern Recognition (ICPR 2018), pp. 2618-2623 (2018).

[13] K. Kato, A. Takefusa, H. Nakada, and M. Oguchi, Construction Scheme of a Scalable Distributed Stream Processing Infrastructure Using Ray and Apache Kafka, in Proceedings of the 34th International Conference on Computers and Their Applications (CATA 2019), pp. 368-377 (2019).

[14] Apache Hadoop, available at https://hadoop.apache.org/ (accessed June 1, 2019).

[15] Apache Storm, available at http://storm.apache.org (accessed June 1, 2019).

[16] Apache Flink, available at http://flink.apache.org (accessed June 1, 2019).

[17] Apache Spark Streaming, available at http://spark.apache.org/streaming/ (accessed June 1, 2019).

[18] M. A. Lopez, A. G. P. Lobato, and O. C. M. B. Duarte, A Performance Comparison of Open-Source Stream Processing Platforms, in Proceedings of 2016 IEEE Global Communications Conference (GLOBECOM 2016), 6 pages (2016).

[19] J. K. Liu, M. H. Au, W. Susilo, K. Liang, R. Lu, and B. Srinivasan, Secure Sharing and Searching for Real-Time Video Data in Mobile Cloud, IEEE Network, Vol. 29, No. 2, pp. 46-50 (2015).

[20] J. Wang, B. Amos, A. Das, P. Pillai, N. Sadeh, and M. Satyanarayanan, Enabling Live Video Analytics with a Scalable and Privacy-Aware Framework, ACM Transactions on Multimedia Computing, Communications, and Applications, Vol. 14, No. 3s, Article No. 64 (2018).

ⓒ 2020 Information Processing Society of Japan