K-008

GPU シェーダを用いた全方位動画の高速再生方式

360-Degree Video Player Using GPU Shader

松本 達弥† 藤田 悟 ‡
Tatsuya Matsumoto Satoru Fujita

1. まえがき

Android スマートフォンに, 搭載された方位センサーに連動し, 表示する領域を変化させる全方位動画の再生システムは, 麻生ら[1]による提案などがある. しかし, これら既存の研究では, 再生機能において,速度や画面同期などの課題があった. そこで, 本論文では, これらの課題を解決する新たなスマートフォン上での全方位動画再生手法を提案する.

2. スマートフォンによる全方位動画の概要と課題

全方位動画の撮影方法には, 複数のカメラを利用する方法と, 全方位を写す事のできる特殊なミラーを利用する方法がある. 本論文では, 後者のミラーを利用し, 全方位動画の撮影を行う. 撮影された全方位の映像は, 図 1 のようになる. この同心円状の映像を, 2 次元極座標から 2 次元直行座標への変換式に基づき, パノラマ化した上で, 表示する. しかし, この時, 動画の両端が連続しているように見せるための処理が必要であり, これを如何に, 低負荷に, 無駄なく処理できるかが, スマートフォンによる全方位動画再生を実現する際の課題である.

先行研究では, 動画の両端の接続についての課題を解決するために, 画面上に動画プレイヤを 2 個並べ, 同期を取った上で再生を行う手法, 折り返し部分に, 画面サイズ分, 重なりを持って記録しておく拡張パノラマ手法[1], 動画ではなく, JPEG 形式の画像を連続して表示し, 動画として再生する MotionJPEG 法などが試みられてきた. しかし, いずれも, ストリーミング再生ができない, 音声再生に難がある場合がある, デコード・描画処理に無駄がある, 事前に PC 上でパノラマ化を行う必要があるなど, いくつかの課題があった.

3. 高速動画再生方式

本論文では, 前述した先行研究の課題を解決するために, ３つの手法を組み合わせたものを提案する.

3.1. GPU シェーダによるパノラマ変換

先行研究による方式では, 事前に, 撮影された全方位動画を, 事前に PC 上でパノラマ状に変換しておく必要があった. 一方,本論文での提案方式では, GPU 上で動作するシェーダにより, リアルタイムな変換処理を行った上で描画を行い, これを解決する. 実際には, 端末の画面サイズと等しい大きさの長方形ポリゴンに, シェーダでパノラマ化処理を行ったテクスチャを貼り付けて, 描画する. また, シェーダを利用した描画を行うために, グラフィクス用 API である, OpenGL ES 2.0[3]を利用する. この API は, GLSL ES という言語により, プログラムを記述することで, GPU でのシェーダの動作を定義できる. また, 利用可能なシェーダは, ポリゴンの頂点ごとに処理を行うバーテックスシェーダと, 画素ごとに処理を行うフラグメントシェーダがある. 今回のパノラマ化処理では, 主に後者のフラグメントシェーダを利用し, 変形処理を行う. なお, 動画のフレームデータをテクスチャとして, シェーダに入力するために, SurfaceTexture という Android のクラスを利用する. SurfaceTexture を利用できるのは, Android OS バージョン 3.0 以降である. 変形処理は, 以下の式に基づき行う.

$$rad = tx * 2.0 * \pi$$
$$r = invisibleR + (visibleR - invisibleR) * ty$$
$$x = centerX + aspect * r * \cos(rad)$$
$$y = centerY + r * \sin(rad)$$

ここで, centerX/cemterY は同心円の中心座標, aspect は動画のアスペクト比, tx/ty は描画するポリゴン上のテクスチャ座標, invisibleR は同心円中心部に存在する周囲の風景が映っていない円の領域の半径, visibleR は周囲の風景が写っている円の領域の半径, x/y は入力された全方位動画テクスチャからピクセルを取得する座標である. 各パラメータについては, 配信サーバ上で, 動画に対する円形ハフ変換などを適用し, 計算した結果を動画配信時に動画と一緒に配信する. 具体的には, 図 1 のように, 中心部の縁の円と外周部の淵の円を検出し, 円の中心点と, 両円の半径距離を求める. この処理は, 動画配信開始時に一度だけ行われる. なお, 求められたパラメータに誤差が生じることもあるため, 端末上でも, リアルタイムに変更可能にする.

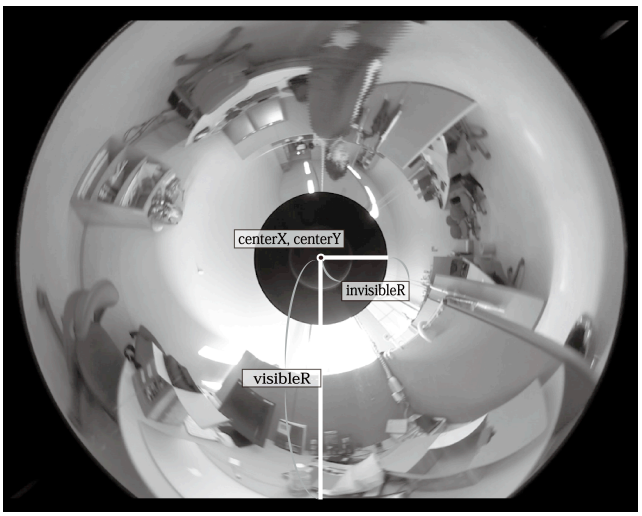

図 1 撮影された動画とパラメータ

3.2 UV アニメーションによるスクロール

提案手法では, 動画の両端が連続しているように表示させるために, 描画時の長方形ポリゴンに対応するテクスチ

† 法政大学 情報科学部 情報科学研究科
‡ 法政大学 情報科学部

ャ座標を変化させる UV アニメーションによりスクロールさせる．これにより，先行研究のように，２つの動画プレイヤを同期し再生させたり，あるいは，拡張パノラマ方式のように，表示時の一画面分，折り返し部分に重なりを持って，動画に映像を記録したりしておく必要がなくなる．よって，同ビットレートの動画に対して，画質が向上し，さらに，デコード負荷・描画負荷が軽減できる．実際には，横スクロールを行うだけなので，U 座標のみを方位センサーに連動させて，値を変化させる．

3.3. HTTP Live Streaming によるストリーミング再生

提案手法では，Android の MediaPlayer クラスを利用し，HTTP Live Streaming プロトコルによるストリーミング再生を行う[3][4]．このプロトコルは，動画を H.264(Baseline Profile)形式，音声を AAC，もしくは，MP3 形式で配信を行うことができることから，先行研究で課題であった，音声再生についても解決できる．このプロトコルは，Apple 社のスマートフォン向け OS である iOS などでも利用可能な方式であり，将来的に，他プラットフォームで，動作させる際の互換性の点を考え，今回は，このプロトコルを採用した．

なお，本論で提案する全方位動画プレイヤでは，ローカルストレージからや，ネットワークからのストリーミングではない配信(プログレッシブ配信)も利用可能である．これらも含めた再生機能の実装については，Android の OS バージョン毎の MediaPlayer クラスの実装に依存する．この MediaPlayer の動画コーデック・配信プロトコルへの対応レベルは，OS のバージョンによって異なり，HTTP Live Streaming プロトコルを利用できるのは，Android OS バージョン 3.0 以降に限定される．

4. 実装

提案手法を実現するために，以下の開発環境で，検証用システムを作成した．ローカルストレージ，あるいは，ネットワークから HTTP Live Streaming プロトコルにより，動画データを受信し，再生を行う．また，システムの概要は図 2 に示すとおりである．

開発環境： Android SDK r18 (Google Inc.)
Android NDK r8 (Google Inc.)

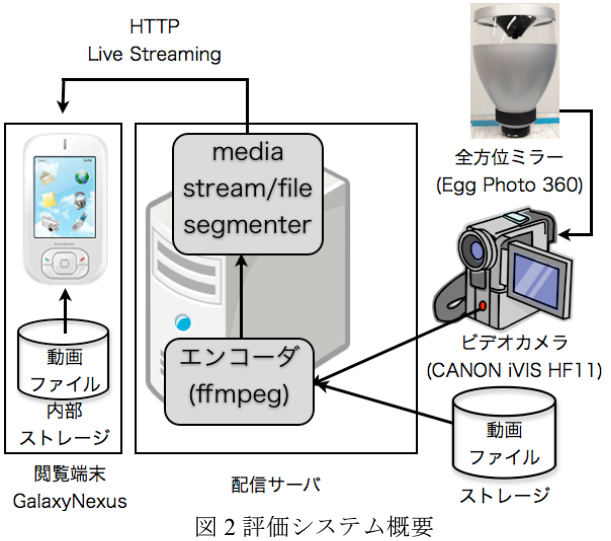

図 2 評価システム概要

5. 評価

機能面から，提案手法と先行研究の手法と比較すると，ストリーミング再生が可能，事前に PC 上で，パノラマ化変換処理を行う必要がない，音声再生が可能という再生機能に関する３点全てが実現できるという点で，本論の提案手法が優れている．しかし，古い OS バージョンへの互換性という点では，提案手法は，Android OS バージョン 3.0 以降でのみ利用可能であることから，先行研究の手法の方が優れている．

性能面・画質面で比較すると，MotionJPEG 以外の手法では，ハードウェア動画デコーダが利用でき，高品質な動画再生が可能である．特に，デコード・描画しなくてはならない領域が，他手法に比較して，少ないことから，画質・デコード負荷・描画負荷いずれも，提案手法による再生が，先行研究の手法のいずれよりも優れていると考える．

	ストリーミング	事前パノラマ	音声再生	互換性
プレイヤ同期	×	×	×	○
Motion JPEG	×	×	×	○
拡張パノラマ	○	×	○	○
提案手法	○	○	○	×

表 1 従来手法との機能比較

6. おわりに

本論文では，スマートフォン上での方位センサーに連動した全方位動画再生方式について，先行研究で課題であった，PC 上での事前パノラマ化，ストリーミング再生，音声再生を GPU シェーダによるリアルタイムなパノラマ変形処理，UV アニメーションによるスクロール，HTTP Live Streaming によるストリーミング再生を実装することで，解決する新たな手法を提案した．そして,提案手法を検証するためのシステムを構築し，実際に，先行研究と比較し，より優れた機能・性能・画質を実現できることを実証した．

今後の課題として，Apple 社の iOS や，Web ブラウザ上での本論での提案方式による全方位動画プレイヤの実装があげられる．これらのプラットフォームでは，細かい点では，Android OS とは異なる．しかし，OpenGL ES 2.0 による描画や，動画をテクスチャとして利用できるなどの要素技術は同一のものが利用できるため，本論の提案方式を応用し，利用することが可能であると考える．

7. 参考文献

[1] 麻生祐，谷口由佳，藤田悟，“方位センサと連動した全方位動画再生方式”，電気情報通信学会，第 4 回ヒューマンプローブ研究会，Sep 2011.

[2] The Khronos Group Inc. "OpenGL ES 2.0 Reference Pages ". OpenGL ES Software Development Kit. (オンライン). http://www.khronos.org/opengles/sdk/docs/man/. (参照 2012-06-28)

[3] Google Inc. "Android Deveropers ". Android Developers. (オンライン). http://developer.android.com (参照 2012-06-28)

[4] Apple Inc. "HTTP Live Streaming の概要". iOS Dev Center. (オンライン). https://developer.apple.com/jp/devcenter/ios/library/documentation/StreamingMediaGuide.pdf. (参照 2012-06-28)