

# GPU を用いたパノラマ画像の生成

小野 貴大 田井 大貴 千神 将太

## 要約

本研究では、パノラマ動画画像を用いたビデオチャットシステムを提案する。ここで言うビデオチャットとは、Web カメラとマイクを用いた PC 間で映像や音声のやりとりである。現在、ビデオチャットで用いられている Web カメラは単眼カメラであり、映像が映る範囲が狭いという欠点がある。そこで、Web カメラ 2 台を使用し、それぞれのカメラから取得する画像を合成することで得られるパノラマ画像を用いることで、広画角な画像を利用するビデオチャットを実現する。パノラマ画像の生成方法は、各カメラから画像を取得後、それぞれの画像に対して射影変換処理を施し、最後に変換した画像同士を合成するというものである。また射影変換とは、図形の直線性を維持しながら画像に遠近感を持たせる変換方法である。

ビデオチャットはリアルタイムで映像を映すため、CPU で処理を行う場合多大な負荷を強いることになり、市販の Web カメラのリフレッシュ周期内に処理を完了することができない。そこで、本研究では、現在の PC に搭載されているグラフィックスハードウェアである Graphics Processing Unit (以下、GPU と略記) を汎用的に用いる技術 General Purpose computing on GPU (以下、GPGPU と略記) に着目し、GPGPU を利用する並列計算アーキテクチャである Compute Unified Device Architecture (以下、CUDA と略記) に射影変換処理を実装する。

This research introduces a video chat system using panoramic images produced by reading available hardware. In order to build this system, we investigated a method and technique of implementing panoramic image formation processing in CUDA, i.e. the parallel computing architecture of GPGPU that can be used for general purposes. The panoramic image formation processing we used here combines two pictures taken separately by two web cameras into one, after they are transformed by projective geometry.

## キーワード

GPU, GPGPU, CUDA, パノラマ画像

GPU, GPGPU, CUDA, panoramic image

## 1 まえがき

現在、情報通信技術の発展にともない、コミュニケーション技術が多様化している。そのコミュニケーション技術の 1 つにビデオチャットがある。ビデオチャットとは、Web カメラやマイクなどを使った PC 間での映像および音声の相互通信である。既存のビデオチャットは、Web カメラ 1 個を用いるため、Web カメラを通して写る映像の範囲に限られる。

複数枚の画像を合成することにより、より広画角な画像を実現するパノラマ画像がある[1]。本研究では、2 個の Web カメラを使用し、それぞれのカメラが映し出す映

像に合成処理を施すことで、パノラマ動画画像を用いたビデオチャットシステムを提案する。

ビデオチャットはリアルタイムな映像を必要とするため、Web カメラのリフレッシュ周期内にパノラマ画像の生成処理を終了しなければならない。しかし、CPU によって動画画像の変換処理を行うことは、CPU に多大な負荷を強いることになり、パノラマ画像の合成処理を Web カメラのリフレッシュ周期内に終了することができない。そこで、現在の PC にビデオカードと呼ばれる専用のグラフィックスハードウェアが搭載されていることに着目し、CPU の代わりにこのハードウェアを利用する。ビデオカードには

画像処理プロセッサである Graphics Processing Unit (以下, GPU と略記) が内蔵されており, GPU を汎用目的で利用する技術として, General Purpose computing on GPU (以下, GPGPU と略記) [2] [3] [4] という技術がある. そこで, 本研究では, パノラマ画像の合成処理を, GPGPU を用いた並列計算アーキテクチャである CUDA に実装する手法について検討する.

以下, 2 章でパノラマ動画の生成法の概要を説明する. 3 章では, 射影変換処理の CUDA への実装方法について述べる. 4 章では, 射影変換処理の CUDA への実装結果について述べる.

## 2 パノラマ動画の生成法

### 2.1 パノラマ動画の生成手順

本章ではパノラマ動画の生成手順について述べる. 以下にパノラマ動画の生成手順を示す.

1. 各カメラから画像を取得
2. 取得画像の変換処理
3. 各画像の重ね合わせ

※1 から 3 を毎フレーム繰り返す

2 では, 座標変換処理として射影変換, グラデーション処理として, 合成時における境界付近の輝度調整を行う.

3 では, カメラ毎に処理された画像を重ね合わせる.

### 2.2 射影変換

射影変換とは, 直線性を維持しながら矩形を任意の四角形に変形することで, 画像に遠近感を持たせる変換である. 射影変換式は次式で表される.

$$\begin{aligned} X(x, y) &= \frac{m_0x + m_1y + m_2}{m_6x + m_7y + m_8} \\ Y(x, y) &= \frac{m_3x + m_4y + m_5}{m_6x + m_7y + m_8} \end{aligned} \quad (1)$$

ここで,  $(x, y)$  および  $(X, Y)$  は, 変換前および変換後の座標,  $m_0 \sim m_8$  は射影変換のパラメータである. 変換先の座標を変換元の座標から正変換で求める場合, 計算された座標の間隔が広いと, 生成された画像に隙間が生じたり,

逆に間隔が狭い場合は同じ画素が重複してしまうという問題がある. そこで, 座標の変換を行う場合には, 変換後の座標に対応する変換前の座標を算出する逆変換式が必要になる. 式(2)に示す逆変換式は, 式(1)を  $x, y$  について解くことで求めることができる. 逆変換式を以下に示す.

$$\begin{aligned} X(x, y) &= \frac{m'_0x + m'_1y + m'_2}{m'_6x + m'_7y + m'_8} \\ Y(x, y) &= \frac{m'_3x + m'_4y + m'_5}{m'_6x + m'_7y + m'_8} \end{aligned} \quad (2)$$

ここで, 逆変換式のパラメータ  $m'_0 \sim m'_8$  は次のようになる.

$$\begin{aligned} m'_0 &= m_4m_8 - m_7m_5 \\ m'_1 &= m_7m_2 - m_1m_8 \\ m'_2 &= m_1m_5 - m_4m_2 \\ m'_3 &= m_6m_5 - m_3m_8 \\ m'_4 &= m_0m_8 - m_6m_2 \\ m'_5 &= m_3m_2 - m_0m_5 \\ m'_6 &= m_3m_7 - m_6m_4 \\ m'_7 &= m_6m_1 - m_0m_7 \\ m'_8 &= m_0m_4 - m_3m_1 \end{aligned}$$

なお, 射影変換パラメータ  $m_0 \sim m_8$  はカメラ配置によって決まる.

### 2.3 射影変換パラメータの算出法

射影変換パラメータの算出方法について述べる. 図 1 に撮影面と投影面の関係を示す. ここで,  $XY$  及び  $xyz$  は, 視点座標系及び撮影面座標系である. 直行座標系である  $xyz$  を極座標  $(r, \theta_a, \theta_b)$  で表し, 極座標  $(r, \theta_a, \theta_b)$  上に,  $x$  軸と  $z$  軸が交わり,  $z$  軸が原点  $O$  を通るように球面上に設置する. 撮影面は投影面と  $Y$  軸に対して  $\theta_b$  傾いているので, 撮影面の  $y$  軸を中心に反時計回りに  $\theta_b$  回転させる. これにより,  $xy$  平面と  $XY$  平面は平行となる. 次に  $z$  軸から見下ろした場合に,  $x$  軸は  $X$  軸に対し,  $\theta_a$  傾いているので, 撮影面の  $z$  軸を中心に反時計回りに  $\theta_a$  回転させる. これにより,  $x, y$  軸と  $X, Y$  軸の方向は一致する. 最後に  $X, Y, Z$  軸方向にそれぞれ平行移動させることで撮影面座標系と視点座標系を一致させる.

次に、この投影面の座標に対して投影変換を行うことで、射影変換パラメータである  $m_0 \sim m_8$  は以下のようになる。ここで、原点  $O$  から撮影面  $Op_1$  までの距離を  $\overline{OOp_1}$  とする。

$$\begin{aligned}
 m_0 &= \cos \theta_a \cos \theta_b \\
 m_1 &= -\sin \theta_a \\
 m_2 &= -r \cos \theta_a \sin \theta_b \\
 m_3 &= \sin \theta_a \cos \theta_b \\
 m_4 &= \cos \theta_a \\
 m_5 &= -r \sin \theta_a \sin \theta_b \\
 m_6 &= \frac{\sin \theta_b}{\overline{OOp_1}} \\
 m_7 &= 0 \\
 m_8 &= \frac{r \cos \theta_b}{\overline{OOp_1}}
 \end{aligned}$$

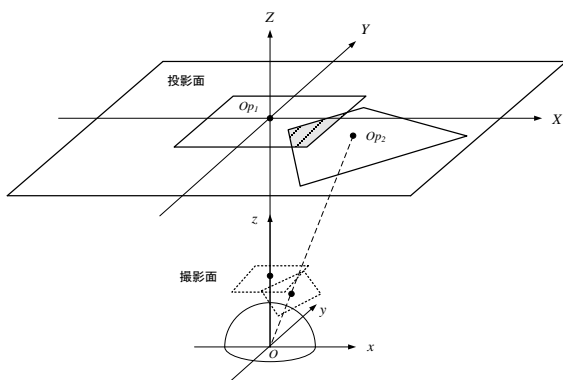


図1 撮影面と投影面の幾何学的関係

### 3 射影変換処理のCUDA への実装

#### 3.1 Compute Unified Device Architecture

2. で述べたパノラマ画像の生成における演算処理をGPU を用いて実行する技術として、GPGPU がある。GPGPU を利用する並列計算アーキテクチャとして、Compute Unified Device Architecture (以下、CUDA と略記) が広く利用されている。

CUDA のプログラム構成のイメージ図を図 2 に示す。CUDA では、CPU 側をホスト及びGPU 側をデバイスと呼び、それぞれが管理するメモリをホストメモリ空間及びデバイスメモリ空間という。CUDA のプログラムはGPU を動作させるデバイスコードと CPU を動作させるホストコード

から構成されている。デバイスで動作する関数をカーネル関数と呼び、ホストコードにはホストで動作するプログラムと、カーネル関数の動作実行を命令するカーネル関数コールを記述する。

CUDA はデバイスメモリ空間にあるデータを利用して演算を行うため、カーネル関数コールの前に、ホストメモリ空間からデバイスメモリ空間にデータ転送を行う必要がある。また、ホストはデバイスメモリ空間にアクセスできないため、演算結果を表示する場合には、カーネル関数の終了後、デバイスメモリ空間からホストメモリ空間にデータ転送を行う必要がある。

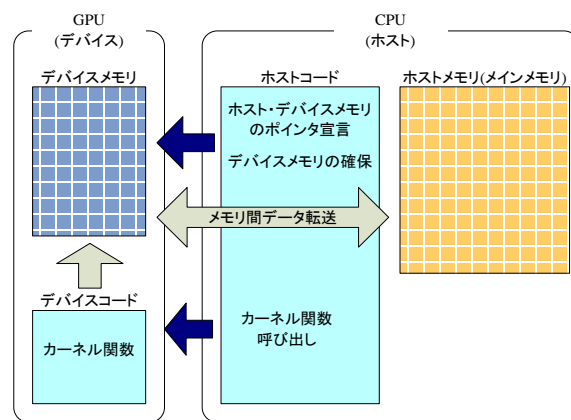


図 2 CUDA のプログラム構成のイメージ

以上から、CUDA の動作手順は以下ようになる。

1. ホストメモリ空間にデータを用意
2. デバイスメモリ空間にデータ領域を確保
3. ホストメモリ空間からデバイスメモリ空間にデータ転送
4. カーネル関数を呼び出し
5. デバイスメモリ空間からホストメモリ空間に演算結果を転送
6. 演算結果を表示

#### 3.2 射影変換手順

射影変換の際の変換原理は、2. で述べた原理に従う。CUDA を用いた射影変換処理の概略手順を以下に示す。

1. ホストで、原画像情報を元に射影変換処理後の画像サイズを算出
2. ホストメモリ空間に変換後の画像情報を格納するための領域を確保

3. デバイスメモリ空間に原画像情報を格納するための領域を確保
4. デバイスメモリ空間に変換後の画像情報を格納するための領域を確保
5. 原画像情報をホストメモリ空間からデバイスメモリ空間に転送
6. GPU によって変換処理を実行
7. 変換処理後の画像情報をデバイスメモリ空間からホストメモリ空間に転送
8. 変換結果を出力

手順 1 では、原画像の 4 隅の画素に対して射影変換処理を施すことにより、それぞれの画素の変換後の座標を決定する。そして、対角に位置する画素の  $x$  及び  $y$  座標の差を求めることで、射影変換後の画像のサイズを決定する。この処理を行う関数のフローチャートを図 3 に示す。

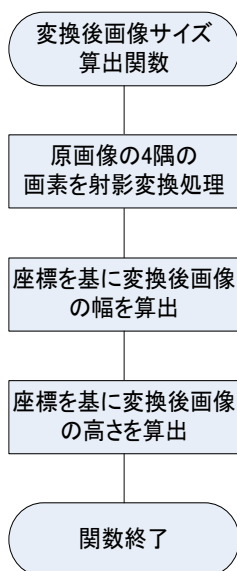


図 3 変換画像サイズ算出関数

手順 6 では、GPU によって全ての画素に対して並列に変換処理を行う。ここで、変換処理には逆射影変換式を用いる。この処理を行う関数のフローチャートを図 4 に示す。

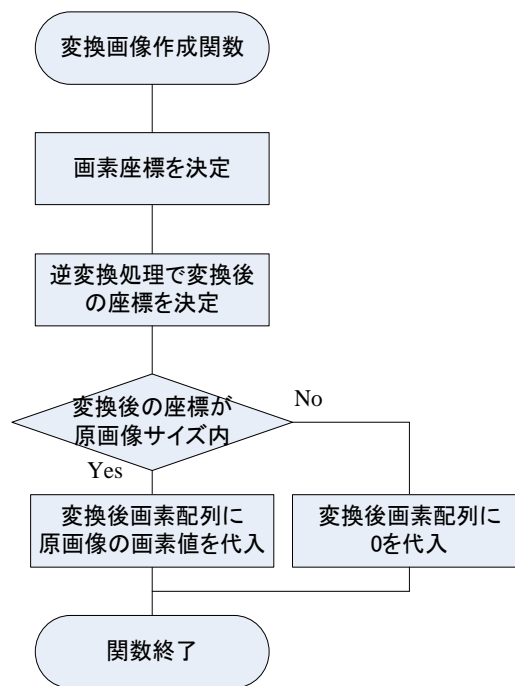


図 4 変換画像作成関数

#### 4 考察

CPU および GPU のそれぞれにおける射影変換処理の実行時間の測定を行った。測定に使用した環境を以下に示す。

- CPU : Pentium Dual core E6800
- GPU : NVIDIA 社製 GeForce GTX460
- OS : Windows 7 64bit
- 原画像解像度 : 640×480
- 変換後画像解像度 : 1043×930

図 5 に変換前の画像を、図 6 に変換後の画像を示す。GPU を用いて処理を行った際の実行時間及び CPU を用いて処理を行った際の実行時間の測定結果を表 1 に示す。GPU で処理を行った方が高速である。

画面更新に関する指標に、単位時間当たりに表示されるフレーム数を表す frames per second (以下、fps と略記) がある。今回の測定結果は 24fps (GPU で処理) であり、市販の Web カメラの 30fps 以内に処理を終えることができていない。よって、合成処理に必要な時間を考慮し、更なる高速化が必要である。



図5 変換前画像



図6 変換後画像

表1 測定結果(msec)

	GPU	CPU
実行時間	41.5	47.0

## 5 あとがき

本研究では、射影変換処理のCUDAへの実装を行った。

画像に対する射影変換処理は、4隅の画素に対して射影変換処理を行い、画像サイズを決定したあと、変換後の画像の全ての画素を逆射影変換することによって、原画像の画素の座標を求め、変換後の画素情報を決定した。また、実装した射影変換処理の実行時間を測定した結果によれば、GPUを用いて変換処理を行った場合、市販のWebカメラのリフレッシュ周期内に射影変換処理を完了できていないため、高速な射影変換を可能とするアルゴリズムに変更する必要がある。

実装したシステムに合成処理機能を加え、射影変換の更なる高速化を目指すことが今後の重要な課題である。

## 6 謝辞

本論文を作成するに際し、有益なご指導ご助言を賜りました岡山県立大学の佐藤洋一郎先生に、また本研究を進める過程で同じく有益なご助言を賜りました川崎医療福祉大学の近藤真史先生に衷心より感謝申し上げます。また授業を通じてご指導を戴きました谷野一忠先生ならびに森洋史先生に対しましても、ここに感謝の意を申し上げます。

## 7 参考文献

[1] 部分広角動画像の提示を前提とした全集動画像の生成に関する研究

木村康秀

平成19年度 岡山県立大学大学院修士論文

(2007-2)

[2] はじめてのCUDAプログラミング

青木尊之, 額田彰

工学社 (2009-11)

[3] CUDAプログラミング実践講座

David B. Kirk Wen-mei W. Hwu

ボーンデジタル (2009-11)

[4] <http://developer.nvidia.com/> : NVIDIA Developer Zone

[5] 透視化マルチウィンドウシステムにおけるウィンドウ操作機構のCUDAへの実装

森洋史

平成23年度 岡山県立大学卒業論文

(2012-2)