パノラマ画像生成処理の CUDA への実装

藤井　壱成　　芦野　祐太
指導教員　谷野　一忠

要約

本研究では，パノラマ動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 間での映像および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，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 によって画像処理を行う場合，多大な負荷を強いることになる．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) に射影変換処理及び合成処理を実装する ．

Abstract

“Video chats”, have become very popular and people can now talk face to face on it. In this system, a single Camera Image is commonly used, but the view is quite narrow.
To make such a system more useful, we suggest a system using composite panoramic images through projective transformation using GPU.

キーワード

GPU，ビデオチャット，パノラマ画像，CUDA

Keywords

GPU, Video Chat, Panoramic Image, CUDA

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が限られる．

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る．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 章でパノラマ動画像の生成法の概要を説明する．3 章では，パノラマ画像生成処理の CUDA への実装方法について述べる．4 章では，パノラマ画像生成処理の CUDA への実装結果について述べる．

2 パノラマ動画像の生成法

2.1 パノラマ動画像の生成手順

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

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．

$$X(x,y)=\frac{m_0x+m_1y+m_2}{m_6x+m_7y+m_8}$$

$$Y(x,y)=\frac{m_3x+m_4y+m_5}{m_6x+m_7y+m_8} \tag{1}$$

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～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，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．

$$X(x,y)=\frac{m'_0x+m'_1y+m'_2}{m'_6x+m'_7y+m'_8}$$

$$Y(x,y)=\frac{m'_3x+m'_4y+m'_5}{m'_6x+m'_7y+m'_8} \tag{2}$$

ここで，逆変換式のパラメータ m'_0～m'_8 は次のようになる．

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

なお，射影変換パラメータ m_0～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．図 2 に示すように，直行座標系である xyz を極座標 (r,θ_a,θ_b)で表し，極座標(r,θ_a,θ_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 θ_b 傾いているので, 撮影面の y 軸を中心に反時計回りに θ_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，θ_a 傾いているので，撮影面の z 軸を中心に反時計回りに θ_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～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．

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

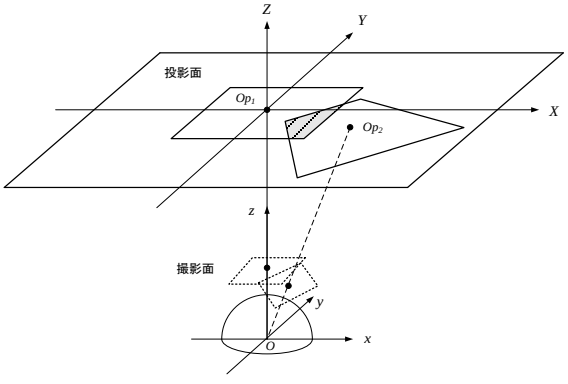
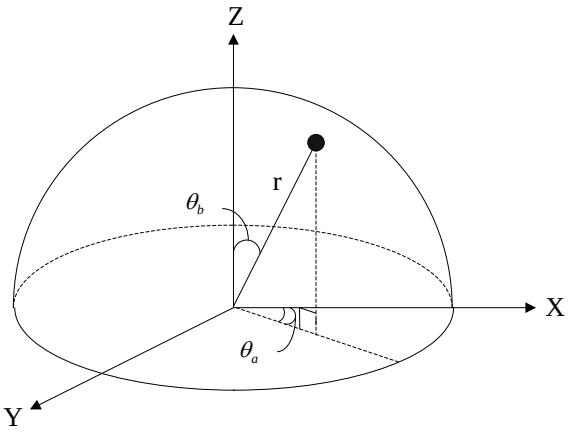
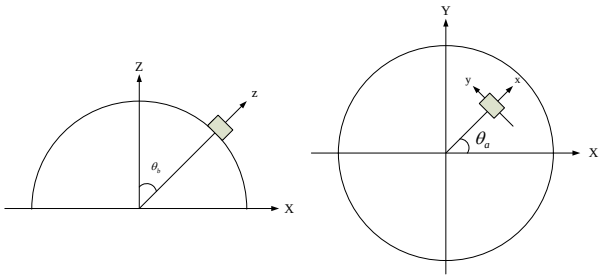


図 1　撮影面と投影面の幾何学的関係



(a)カメラ位置の極座標



(b),(c)カメラの位置

図 2　極座標における (b),(c)カメラの方向

2.4 合成

1.で述べた2台のwebカメラを用いてパノラマ画像を生成する方法について述べる．webカメラの配置図を図3に示す．WebカメラはPCの使用者に対して同じ角度分逆方向に傾ける．

双方の画像に対し，それぞれの角度によって求められたパラメータを用いて射影変換処理を施すことにより，各画像の内側が重なり，一枚のパノラマ画像を生成する．

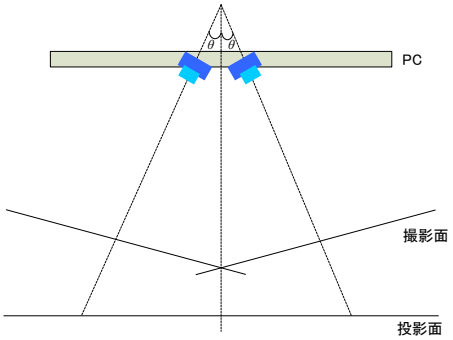


図 3

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 4 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．

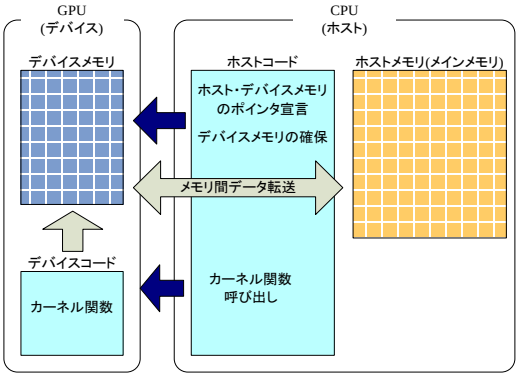


図 4　CUDA のプログラム構成のイメージ

以上から，CUDA の動作手順は以下のようになる．

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・合成手順

射影変換および合成の際の変換原理は，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. ホストメモリ空間とデバイスメモリ空間をマップ
6. GPU によって変換処理を実行
7. 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 5 に示す．

手順 5 では，マップドメモリによってホストメモリ空間とデバイスメモリ空間をマップすることによりデータ転送を行う

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 6 に示す．

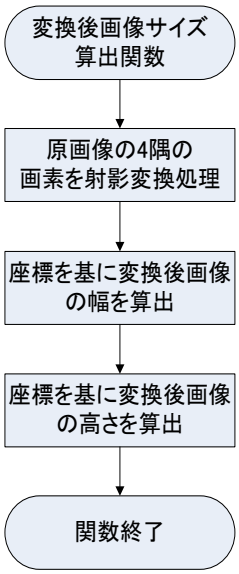


図 5　変換画像サイズ算出関数

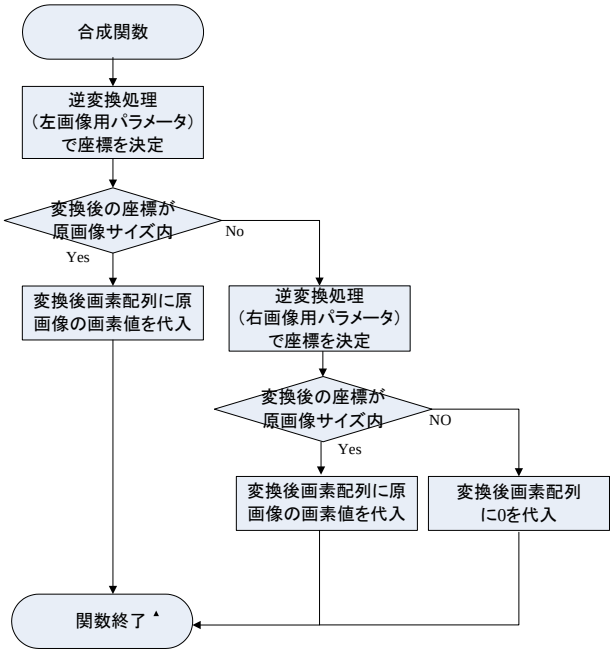


図 6　合成関数

まず、左画像用のパラメータを用いて逆射影変換をおこなう．変換後の座標が左画像のサイズ内であれば，その座標の画素値をコピーする．原画像内でなければ，右画像用のパラメータを用いて逆射影変換を行う．同様に，右画像のサイズ内であれば，その座標の画素値をコピーし，そうでなければ黒の画素値を代入する．

4 考察

CPU および GPU のそれぞれにおける射影変換処理及び合成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：1316×546

図 7 に合成前の画像を，図 8 に合成後の画像を示

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ことがわかる．

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 62fps(GPU で処理)であり，市販の Web カメラの 30fps より大きいため，本研究の有用性を示すことができた．

(a)左カメラ用画像　　(b)右カメラ用画像

図 5 合成処理前画像

図 6　合成処理後画像

表 1　測定結果(ms)

	GPU	CPU
実行時間	16	47

5 あとがき

本研究では，パノラマ画像生成処理の CUDA への実装を行った．

2 枚の画像に対する合成変換処理は，左画像の左上，左下，および右画像の右上，右下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，実装した合成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より高い値で合成変換処理を完了できることが明らかとなった．

実用化に向けて研究を進めていく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)

[6] 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
2012 年度　金光学園高等学校　探究Ⅱ論文 (2012-11)