コンピュータ理工学特別研究報告書

題目

2台の360°カメラを用いた立体360°画像の生成に関する研究

学生証番号　644952

氏名　西島　啓斗

提出日　令和2年1月22日

指導教員　蚊野　浩

京都産業大学
コンピュータ理工学部

要約

Oculus など比較的安価な HMD の登場によって，VR コンテンツを体験する機会が増えた．HMD で鑑賞するコンテンツの一つに，立体 360°画像がある．実写の立体 360°画像は，テレビで見る映像と比較して，没入感が高くリアルな体験をすることができる．現在，立体 360°画像を撮影するためには，高価で大掛かりな装置が必要である．そこで，より手軽に，実写 VR コンテンツを作成できるようなシステムを研究したいと考えた．

本研究では，全周囲で立体視可能な 360°画像を手軽に撮影できるシステムを作成することを目標とした．そのために，撮影機材として 2 台の 360°カメラを使用した．2 台の 360°カメラを並べて撮影した画像をそのまま観察すると，正面方向以外では正しく立体視をすることができない．2 枚の画像に生じる視差をグラフ化して，どのような処理を行うべきか考えた．すると，後ろ方向では視差が反対になっていること，横方向では水平視差がなく垂直視差のみが生じていること，などがわかった．そこで，撮影した画像に対して，

① 隣のカメラを消去する
② 画像を入れ替えて対応点でズレを修正する
③ 最小二乗法を用いて色をあわせる
④ 画像の継ぎ目を滑らかに変化させる

といった処理を行うプログラムを作成して適用した．結果として，カメラの後ろに当たる部分では視差が逆転して，正しい視差となった．また，横方向に当たる部分では垂直視差がゼロになり，周辺部でも垂直視差が減少した．実際に HMD で観察すると，前方向と後ろ方向では立体感を感じられた．また，横方向でも物が二重に見える事はなく，見え方が改善されていた．しかし，横方向では水平視差がないため，立体視はできなかった．横方向でも立体視を可能にするのが今後の課題である．

目次

1章 序論

Oculus Rift など比較的安価な HMD（Head Mounted Display）の登場によって VR（Virtual Reality，仮想現実感）を体験できる機会が増えた．HMD を使って鑑賞する VR コンテンツに 3DCG を用いたゲームや実写画像などがある．その中でも実写の立体 360°画像は，テレビで見る画像と比較して，格段に没入感が高くリアルな体験が可能になる．

RICHO 社が販売している THETA のような 360°カメラの登場によって，実写の 360°画像を，容易に撮影できるようになった．THETA は魚眼レンズを 2 つ使って，ワンショットで 360°画像を撮影するカメラであり，自分の周囲の景色すべてを記録することができる．HMD を用いて 360°画像を鑑賞すれば，それだけで没入感のある体験が可能である．360°画像に立体感を加えると，さらにリアルな体験が可能になる．現在，立体 360°画像を撮影するには何台ものカメラを組み合わせた機器を使用する必要がある．このような機器は大掛かりかつ高価で，一般人には使うことが難しい．そこで，より手軽に立体 360°画像を作成できる簡易な撮影システムを研究しようと考えた．

人間は 2 つの目で少し離れた位置から世界を見ている．2 つの網膜に生じる像に若干のズレが生じ，このズレを脳内で処理することによって立体感を感じる．このしくみは両眼立体視と呼ばれ，人間が世界を立体的に感じる要因の一つである．平面画像を立体的に見るには，少しズレのある 2 枚の画像（ステレオ画像）を用意する必要がある．このような画像を撮影するためのカメラは，ステレオカメラと呼ばれる．これは，2 台のカメラを横方向に少し離して配置した装置である．ステレオカメラで撮影すれば，人間の目が見ているものと同じようなズレを再現できる．さらに，2 台のカメラの中心を軸として装置を回転させながら多数の画像を撮影し，それらの画像を適切につなぎ合わせれば，全方向で立体視できる画像（立体 360°画像，ステレオ 360°画像）を作ることができる．

通常のステレオカメラと同じように，THETA を 2 台並べて撮影すれば，立体 360°画像を記録できるように思われる．しかし実際には，視線の方向によって画像がずれる方向が異なるため，期待した立体 360°画像にはならない．この問題を画像処理によって解決すれば，全方向で立体視できる画像が作成できると考えた．本研究の目標は，固定した 2 台の 360°カメラで撮影した画像から，全周囲で立体視可能な 360°画像を作成することである．そのために，元の画像に生じている問題を画像処理によって解決する．以下，2 章で立体 360°画像について述べ，3 章で実際の画像の作成手順を説明する．4 章で作成した画像の評価を行い，5 章で結論を述べる．

2 章 立体 360° 画像

人間がシーンに立体感を感じるとき，遠近法，運動視差，輻輳角などいくつかの手がかりを使っている．本研究では両眼視差に注目する．両眼視差とは，右目と左目でものを見たときの，ものの見え方の違いである．右目と左目でものを見たときに見える位置の違いは，奥行きを知る上での手がかりとなる．両眼視差は立体視をする上で重要な要素である．2 章では，立体 360° 画像を撮影するための既存の方法とこれを観察する装置について説明する．

2.1 立体画像の撮影

人間の両目は水平方向に少し離れている．そのため，2 つの目で少し違う視点から物体を見ている．この画像の違いを脳内で処理することで，立体感を感じている．また，実際の立体物を見なくとも，立体物を異なる視点から撮影した 2 つの画像を提示すれば立体視することが可能である．古典的なものとしてステレオスコープがある．

図 2.1　ステレオスコープ

引用元：https://en.wikipedia.org/wiki/Stereoscope

同じ風景を異なる位置から撮影することによって，画像上に僅かなズレが生じる．このズレを視差と呼ぶ. 視差は撮影している物体までの距離とカメラの間隔によって変化する．撮影している物体までの距離が近いほど視差は大きくなる．また，カメラの間隔が遠いほど，視差は大きくなる．

2 台のカメラを，人間の眼と同じように水平に配置して撮影する場合を考える（図 2.2 左）．このように 2 台のカメラを水平方向だけに移動させて 2 枚の画像を撮影し，それを図 2.1 のようなステレオスコープで観察すると，正しく立体視できる．この時，画像上の視差は水平方向に生じる. そして，カメラの光軸間の距離 d をベースラインと呼ぶ.

図 2.2 右に，斜め方向を撮影した場合を示す．正面を向いていたときと比べ，ベース

ラインが狭くなっていることがわかる．ベースラインが狭くなると，観察したときに感じる立体感は弱くなる．また，視線方向に対して左カメラが前に，右カメラが後ろに配置されている．このような配置になると，画像上の視差は水平方向だけではなく，斜め方向にも発生する．更に，図 2.2 右のオレンジの線はカメラが真横を向いた場合を示している．真横を向いた場合は，ベースラインがゼロになっていることがわかる．ベースラインがゼロの場合，画像中央では視差がゼロになり，立体感を感じない．ただしこの場合は，カメラが前後に配置されているので，画像中央以外ではなにがしかの（水平，垂直，斜めの）視差が生じる．

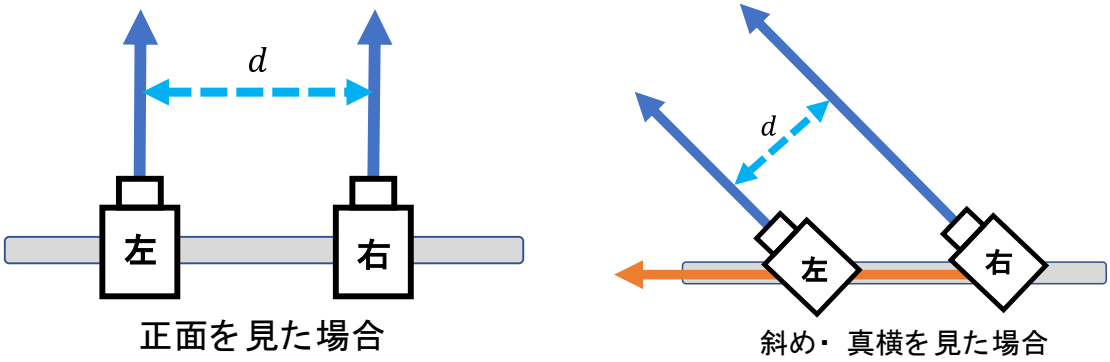


図 2.2　左：カメラで正面を撮影した場合　右：斜め・真横を撮影した場合

2.2 立体 360°画像の撮影に関する従来研究

立体 360°画像を撮影できる既存手法として，2 つのカメラを台に乗せて，回転させながら撮影する方法がある [1]．回転させながら撮影した多数の画像をつなぎ合わせて，360°画像にする．2 つのカメラからそれぞれに 360°画像が得られる．図 2.3 にカメラの様子を示す．この方法では，全方向において，カメラの間隔が保たれているので立体視することができる．

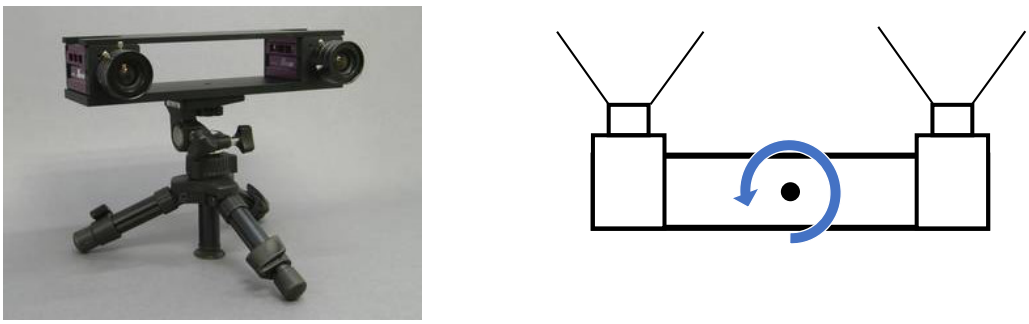


図 2.3　左：ステレオカメラ　右：ステレオカメラを回転させて撮影する様子

引用元：http://www.viewplus.co.jp/product_viewplus_camera/4_index_detail.html

Gopro 社の Odyssey（図 2.4）などでも立体 360°画像を撮影することができる．これはカメラを 10 台以上組み合わせて作られている．カメラを等間隔に配置することによって，全周囲で立体視を可能にしている．この場合も，360°すべての方向でカメラの間隔が保持されているため，立体視できるステレオ画像を撮影することができる．また，この装置では全方向を同時に記録できるので，ビデオを撮影することもできる．

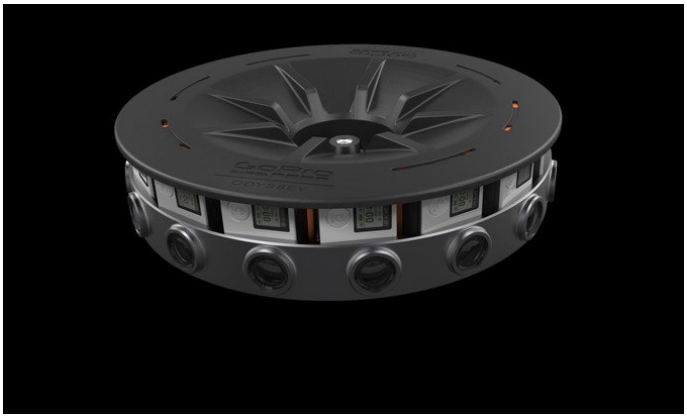


図 2.4　GoPro 社　Odyssey

引用元：https://www.moguravr.com/odyssey-gopro/

2 台の 360°カメラを使って立体 360°画像を生成する研究に[3]がある．[3]は，本研究と同様に，2 枚の 360°画像に画像処理を加えることで，立体 360°画像を生成している．必要な幾何学的変換を厳密に加える方法であるが，提案されている手法を実際に適用して歪みのない立体 360°画像を得ることは難しいと思われる．

2.3 HMD を用いた立体 360°画像の観察システム

本研究のために，立体 360°画像を観察するためのシステムを Unity で作成した．シーン中に，右目と左目それぞれの 360°画像用に，球体オブジェクトを 2 つ配置する．配置した球に，ステレオ 360°画像をテクスチャとして貼り付ける．そして，球の内部にカメラを置いて，HMD の左目，右目それぞれに，対応する画像を映し出す．図 2.5 にプログラムの様子を示す．左側の球に左目用の画像，右側の球に右目用の画像を貼り付けている．

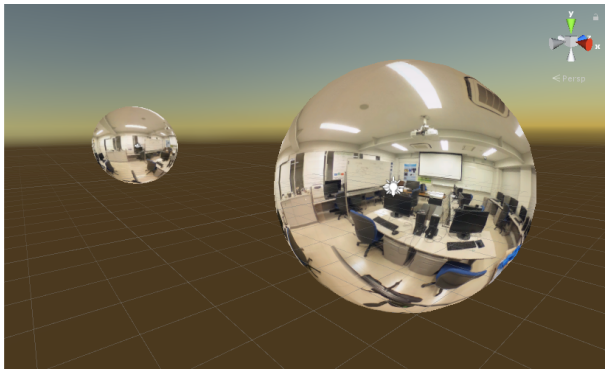


図 2.5 立体 360° 画像を観察する Unity のプログラム

観察するための HMD に Oculus Rift を使用した．Oculus Rift の外観を図 2.6 に，性能を表 2.1 に示す．

図 2.6　Oculus Rift

引用元：https://www.amazon.co.jp/dp/B07KMJK6B8

表 2.1　Oculus Rift の性能

解像度	2160×1200（片目 1080×1200）
リフレッシュレート	90Hz
視野角	水平　90°
	対角　110°

Unity は，標準で VR アプリケーション開発のための基本的な機能を提供している．PlayerSettings→XRSettings の Virtual Reality Supported を有効にすることで，VR アプリケーションの開発が可能になる．設定を有効にすると，自動的に接続した Oculus へ映像が出力されるようになる．また，Unity でアプリを再生したときに，Projection が Perspective に，Field of View が 90° に固定される．

3 章 2 台の 360°カメラを使った立体 360°画像の作成

この章では，2 台の 360°カメラを使って，立体 360°画像を作成するための撮影手順と処理手順を説明する．

3.1 THETA で撮影した 360°画像

THETA は RICHO 社が発売している 360°カメラである（図 3.1）．

図 3.1　RICHO THETA V

引用元：https://theta360.com/ja/about/theta/v.html

THETA は 2 つの魚眼レンズを使用することで，360°をワンショットで撮影できるカメラである．カメラの内部では，図 3.2 のように魚眼レンズ画像が撮影されている．これをカメラ内部で自動的につなぎ合わせて，正距円筒画像として保存している．

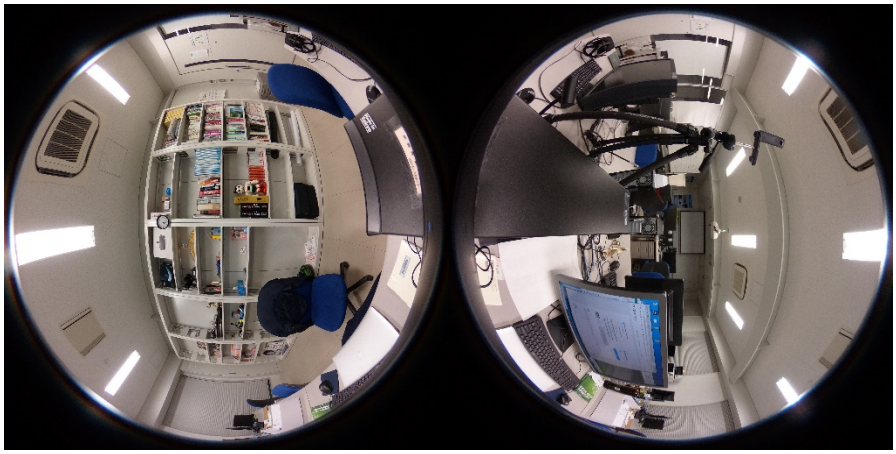


図 3.2　THETA で撮影された魚眼レンズ画像

THETA が撮影する正距円筒画像の性質について説明する．図 3.3 に THETA で撮影した正距円筒画像の例を示す．正距円筒図法とは，球の経度と緯度を，平面画像の縦軸と横軸にそのまま読み替えた図法である．画像中央の水平線上と，縦方向は距離が同じである．一方，画像中央の水平線から上下に離れるほど，物体が横方向に拡大される．

図 3.3　THETA が撮影する正距円筒画像の例

カメラ本体を見て，THETA のロゴが書いてある面を表側，図 3.1 のように撮影ボタンがある面を裏側とする．撮影した正距円筒画像において，カメラの表側正面が画像の中央に対応する．真横側は画像の水平方向 1/4 と 3/4 の位置であり，真後ろは画像の両端部になる．

3.2 THETA を用いたステレオカメラ

2 台の THETA（THETA V と THETA S）を図 3.4 のように据え付けた．間隔を空けてカメラを固定するために，市販の固定治具を用いた．カメラの間隔は治具の最小距離である 95mm とした．カメラを固定した治具を三脚に取り付けて，おおよそ立位時の目の高さまで伸ばして撮影した．治具の両端が写り込んだため，カメラが固定できる幅を残して両端を切り取った．また，三脚が小さく映るように，治具と三脚の間に延長棒を取り付けた．THETA はスマートフォンに Wi-Fi で接続して，リモートで撮影することができる．この機能を用いて 1 台ずつ撮影した．

図 3.4　左：カメラを治具に固定した様子　右：全体の様子

2 台のカメラを同じ設定で撮影した画像を観察すると，色が異なっていることがわかった．色を近づけるために，カメラの設定を変えて撮影した．しかし，色の違いは残ったので，プログラムによる後処理で解決することにした．図 3.5 に，色合いを調整した後の 2 台のカメラで撮影した画像を示す．

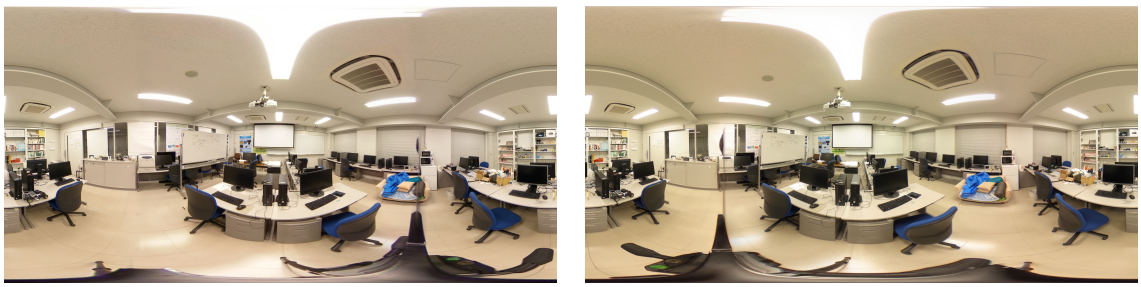


図 3.5　左：左カメラの画像　右：右カメラの画像

理論的には，左右のカメラは完全に水平で撮影面が揃っていることを想定している．しかし，撮影した画像には全体的なズレが少し残っていた．これは，カメラ，治具，三脚の取り付けを手作業で調整していることが主な原因であると考えられる．このズレによって，画像を評価した場合の数値が理論値から偏差を持つことになる．このようなズレは幾何学的な補正を行うことで，取り除くことができると考えられるが，本研究では取り扱わず今後の課題とする．

3.3 2 台の THETA で撮影した 360° 画像の性質

水平方向に視差がついている画像を人間が観察すれば，視差の量に応じて立体感を感じる．しかし，画像の見た目から視差の程度を正確に評価することは困難である．視差の量を定量的に比較するために，THETA で撮影した 2 枚の正距円筒画像の視差を可視化するプログラムを作成した．図 3.5 の画像をこのプログラムに入力したときに，出力されるグラフと画像を図 3.6，3.7，3.8，3.9 に示す．

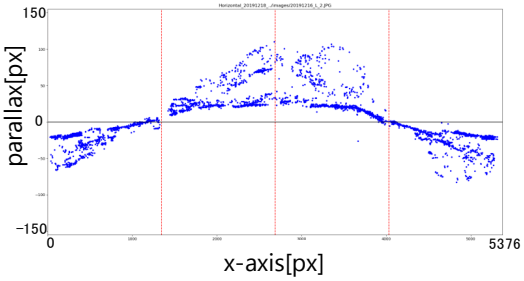


図 3.6　画像の水平位置と
水平視差の関係を示すグラフ

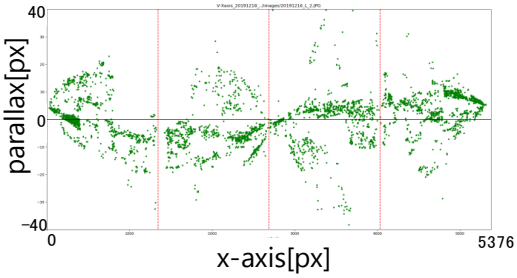


図 3.7　画像の水平位置と
垂直視差の関係を示すグラフ

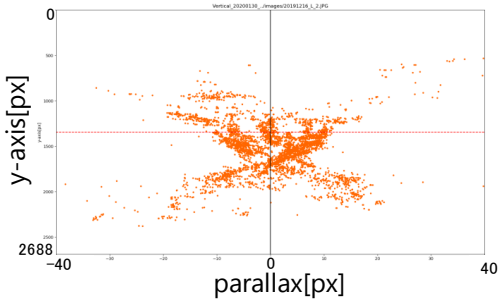


図 3.8　画像の垂直位置と
垂直視差の関係を示すグラフ

図 3.9　視差を可視化した撮影画像

このプログラムは，撮影した 2 枚の画像を入力すると，画像の水平位置と水平視差の関係を示すグラフ（図 3.6），水平位置と垂直視差の関係を示すグラフ（図 3.7），垂直位置と垂直視差の関係を示すグラフ（図 3.8），2 枚の画像を重ねて対応点を線分でプロットした画像（図 3.9），の 4 つを出力する．これらを行うために，画像の特徴点を検出し，マッチングによって同じものが写っている点を探索した．特徴点の計算には A-KAZE 特徴を，特徴点マッチングには Brute Force を使用した．視差量として，左画像の座標から右画像の座標を引いた値を用いた．水平視差は x 座標，垂直視差は y 座標について計算したものである．ここで，Brute Force で得た対応点を目視すると誤った対応点が多く存在した．そこで，対応する点のユークリッド距離とマッチング距離（A-KAZE 特徴の類似度）を設定し，対応点の水平視差が 155 画素より小さく，垂直視差が 80 画素より小さく，マッチング距離が 300 より小さい対応点のみを選択するようにした．図 3.6，3.7，3.9 ではカメラの正面に当たる 1/2，真横に当たる 1/4 と 3/4 の部分に赤線を入れた．図 3.8 の中心にある赤線は，撮影した画像の水平線と対応している．

ここで，2 枚の 360° 画像をどのように処理すれば立体視可能になるかを考えるために，図 3.6 から図 3.9 を精査した．図 3.6 の水平視差のグラフを見ると，画像の真正面では正しい方向に視差が生じている．真横方向を見ると水平視差がゼロである．真横を境界として視差の符号が逆転しており，真裏では正面と逆方向の水平視差が生じている．

図 3.7 の垂直視差のグラフを見ると，真正面では垂直視差はゼロである．横方向に行くにしたがって垂直視差量が増加しており，真横付近で最大になっている．その後，

垂直視差は小さくなり，真裏でゼロになっている．

図 3.8 を見ると，画像の水平線より少し下で，垂直視差は最小になっている．最小になっている部分から，上下に行くほど垂直視差は大きくなっている．視差は正と負どちらの方向にも発生している．

図 3.9 を見ると，図 3.6〜図 3.8 で確認した全体的な傾向を，それぞれの対応点の位置関係から視認できる．カメラの真正面方向では水平視差のみが生じ，真横方法では垂直視差のみが生じ，真裏方向では逆方向の水平視差のみが生じている．それ以外の場所では斜め方向の視差が生じている．正確に言えば，完全にこの傾向に合致しているわけではない．その原因は，2 台の THETA を据え付けた時の誤差や，THETA 自身の幾何学的な歪みによるものと考え，今後の課題とする．

3.4 2 枚の 360°画像を立体 360°画像に変換するための処理

図 3.4 の治具に据え付けた 2 台の THETA で撮影した 2 枚の 360°画像は，理想的な立体 360°画像と比較すると，以下の問題がある．

① 理想的な立体 360°画像になっているのは真正面側だけである．
② 真裏側は，理想的な立体 360°画像と視差が逆転している．
③ 真横方向は，視差が垂直方向にだけ発生している．
④ ①から③以外の一般的な位置では，視差が斜めに発生している．
⑤ それぞれの画像に他方のカメラ，据え付け用の治具，三脚が写り込んでいる．

本研究では，②から⑤を以下の方法で解決あるいは軽減した．②を解決するために，2 枚の 360°画像の間で，カメラの裏側に当たる部分を入れ替える．入れ替えは，水平視差が発生しない真横の位置で行う．この処理を行うと垂直方向に位置ずれした部分が隣り合うことになる．そこで，垂直方向に位置をずらせる幾何学的な変換を施すことで，位置ずれを補正し，また，垂直視差をゼロにする．垂直視差をゼロにすることは本質的な解決ではないが，像が二重に見えるという問題を軽減する．垂直方向に位置をずらせる時，境界以外の領域も，ある程度の変形を受ける．この変形は④の斜め方向の視差を軽減するように働く．他方のカメラが写り込んでいる箇所は，その他方のカメラによって撮影できているので，その画像を変換して埋め込む．据え付け用治具や三脚の写り込みの解決は，将来の課題とする．また，これらのプロセス全体において画像間の色合わせの必要が生じる．

プログラムは C++と Python で作成した．画像処理ライブラリとして OpenCV を用いた．それぞれの処理ごとにプログラムを作成した．処理の流れを図 3.10 に示す．

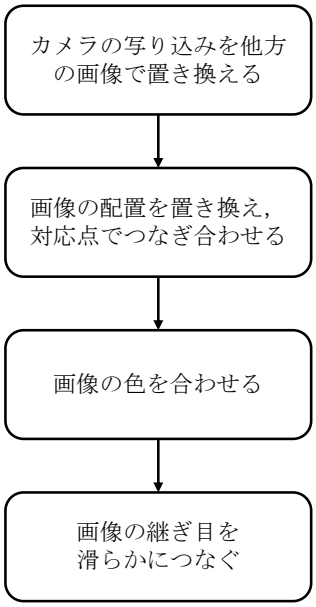


図 3.10　プログラムによる処理の流れ

次の節から，図 3.10 の順に処理の内容を説明する．

3.4.1 カメラの写り込みを他方の画像で置き換える

2 台のカメラで同時に撮影しているので，反対側のカメラが画像に映り込む．これを消すために，反対側のカメラの画像を用いて，カメラの部分を上書きする．図 3.11 に，同じ方向を撮影する 2 つの画像を示す．

図 3.11　左：カメラが写り込んだ右カメラの画像　右：同じ部分の左カメラの画像

写っている物体の大きさや色が, わずかに異なるので, それらを合わせる必要がある. まず, 対応する特徴点の位置を, 視差を可視化するプログラムと同様に, 特徴量検出と特徴点マッチングで求めた. そして, 幾何変換のためのアフィン変換の係数を最小二乗法で求めて, 領域の位置を合わせた. 次に, 対応する画素値から, 画素値を線形変換する式を最小二乗法で求め, 色を合わせた. 以上の処理で求めた画素値を, 画像に上書きした. その際, カメラを消す処理を行った領域の左右の端の, 内側 20 画素において, 元の画素値と処理後の画素値のブレンディングを行い, 領域の継ぎ目を滑らかに変化させた. 図 3.12 に処理の結果を示す.

図 3.12　左：カメラの部分を他方のカメラで上書きした画像
右：色合わせとブレンディングの処理後

図 3.12 の画像を比較すると, 右の画像は, 色の違いが改善して, 継ぎ目も部分的に目立たなくなっていることがわかる. しかし, ブレンディングの処理を行ったにもかかわらず, 継ぎ目が目立っている部分も存在する. 原因として, ブレンディングを行った範囲が 20 画素と狭く, 変化が急になったことが考えられる.

3.4.2 画像を置き換えて対応点でつなぎ合わせる

カメラの裏面で撮影した部分を左右の画像で入れ替えた．正距円筒画像の左側 1/4 と右側 1/4 を入れ替えることになる．図 3.13 に画像を入れ替える様子を示す．

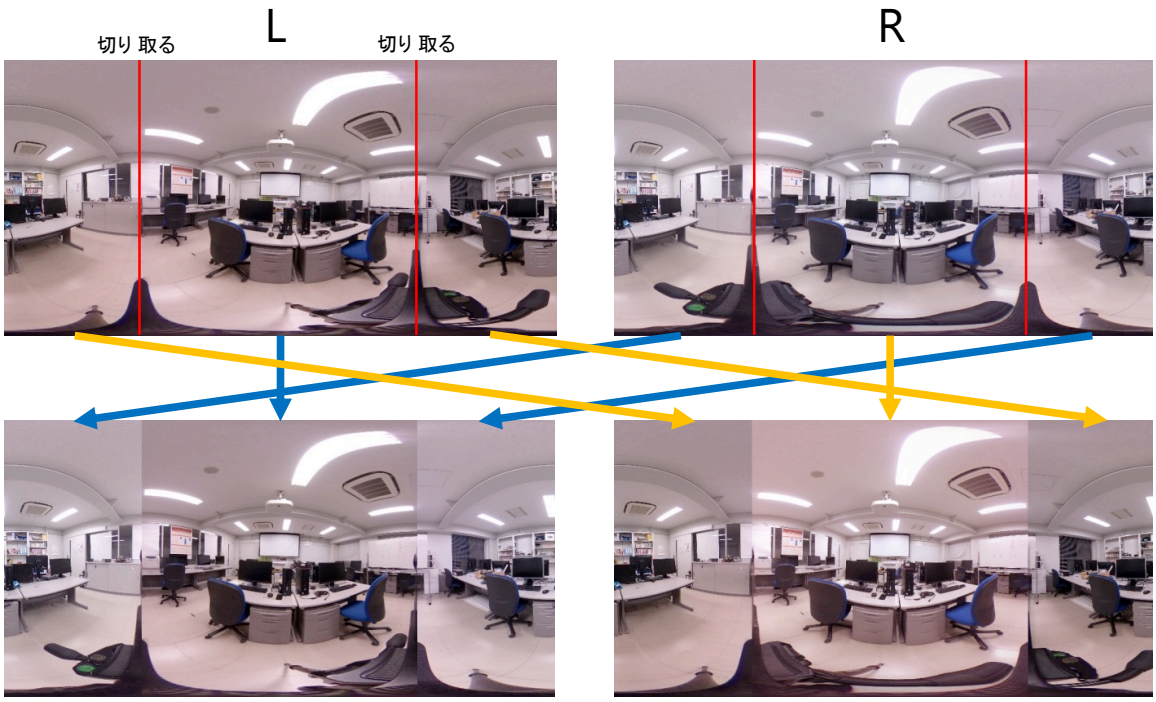


図 3.13　画像入れ替えの様子

図 3.14 に画像を入れ替えた左目用画像を示す．

図 3.14 カメラの裏面で撮影した領域を入れ替えた画像

図 3.14 では，継ぎ目の部分において天井の梁など写っている物体がずれていることがわかる．そこで，画像の対応する点でズレを修正する処理を行った．

画像の修正には，継ぎ目において同じものが写っている対応点が必要になる．画像の対応点は手作業で入力した．その後，対応点ごとに領域を分割して幾何学的な変換処理を加えた．その処理は次のようになる．

① 画像の上端と下端の画素は移動させない．
② 継ぎ目における対応点をその中心に移動させる．
③ 継ぎ目以外の列は，継ぎ目からの距離に応じて移動量を線形補間した．
④ 継ぎ目の反対側（図 3.15 の部分画像 A の左端列，部分画像 B の中央列，部分画像 C の右端列）では画素位置を移動させない．
⑤ 移動させた画素の座標は小数となる事がある．そこで，元の画像から小数点以下の値をもとに，縦方向にのみ画素値を線形補間した．

図 3.15 に画素移動の様子を示す．青色の画素は移動前，オレンジの画素は移動後を示している．

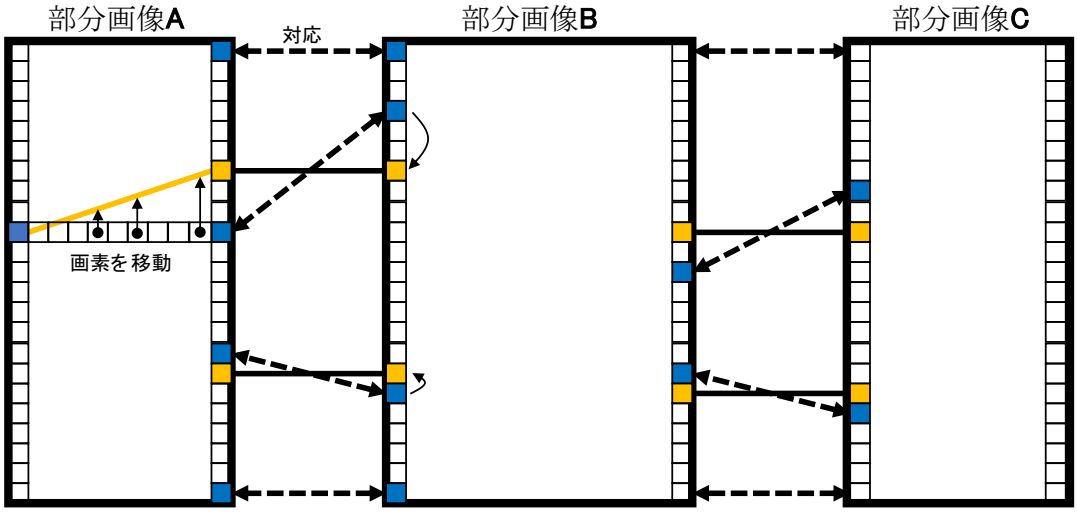


図 3.15　画素の移動の様子

2 台のカメラを使用しているので，撮影される画像の色が微妙に異なる．つなぎ合わせた結果，色の違いが目立つようになった．HMD で観察すると継ぎ目が二重に見える．そこで，色の違いを合わせるプログラムを作成した．継ぎ目における左右の画像の画素値の関係をグラフ化して，概ね線形の関係であることを確認した．濃度変換の式として $V = av + b$ を用い，画像の継ぎ目の画素値から，撮影したカメラごとに係数を求めた．

係数 a,b は最小二乗法で求め，求めた係数での変換を画像全体に適用した．

結果は，適用以前より色が近づいているように見えたが，継ぎ目が見えなくなることはなかった．僅かな色の違いであっても，それを併置すると違いがはっきりする（図 3.16）．継ぎ目が目立たないようにするため，次の項でさらに処理を行う．

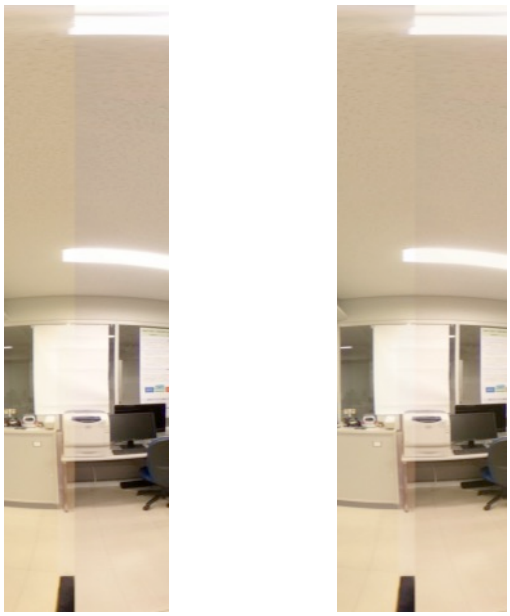


図 3.16　左：色を合わせる前の画像　右：処理を適用した画像

3.4.3 画像の継ぎ目を滑らかに変化させる

HMD で観察したときに画像の継ぎ目が気になった．そこで，継ぎ目が境界として見えないように，継ぎ目の部分を滑らかに変化させた．継ぎ目の左右 100 画素の範囲で，左右の画像を徐々にブレンディングさせた．図 3.17 に処理前と処理後の画像を示す．処理前は境界がはっきりと見えるが，処理後は境界が目立たなくなっていることがわかる．結果として，正距円筒画像を観察したとき，継ぎ目が目立たなくなった．また，HMD で観察したときに，継ぎ目の部分で違和感を覚えることがなくなった．

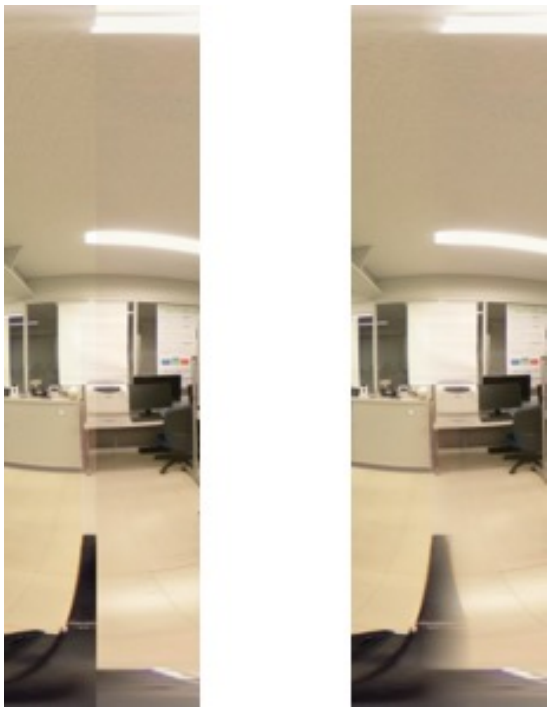


図 3.17　左：ブレンディング前の画像　右：ブレンディング後の画像

作成したプログラムをまとめて実行するために，バッチファイルを作成した．引数として左カメラの画像，右カメラの画像の 2 つを取る．処理が終了すると，左目用画像と右目用画像の 2 つが保存される．

4 章 画像の処理結果と考察

4.1 処理結果

図 3.5 の画像に 3 章の処理を加えて, 立体 360° 画像を生成した. 生成した立体 360° 画像を 3.3 節で説明した水平視差などを可視化するプログラムで処理し, 元の画像からどのように変化したかを定量的に調べた.

図 4.1 に処理を行った前後での水平位置と水平視差の関係を示す. THETA の裏側で撮影した領域は, 処理前にマイナスの視差だったものが, 折り返されてプラスになっている. 正面側と同様の視差となり, 正しく立体視ができるようになったことを意味する.

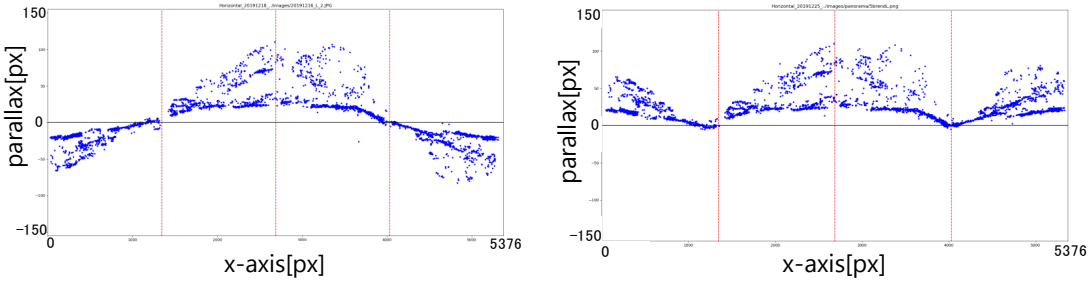


図 4.1　左：処理前の水平視差のグラフ　　右：処理後の水平視差のグラフ

図 4.2 に処理を行った前後での水平位置と垂直視差の関係を示す．カメラの真横に当たる部分で視差がゼロになっている．これは，真横の位置で，画像の特徴点をつなぎ合わせた効果である．また，その周辺でも垂直視差が大きく減少している．

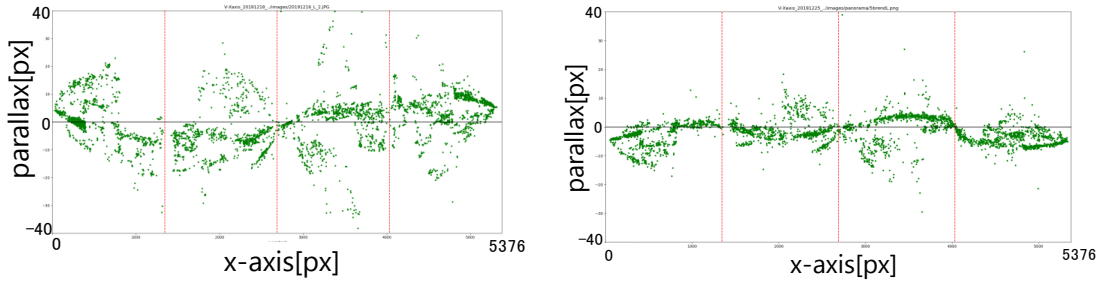


図 4.2　左：処理前の垂直視差のグラフ　　右：処理後の垂直視差のグラフ

図 4.3 に処理を行った前後での垂直位置と垂直視差の関係を示す．図 4.3 では，画像の上下部分で大きくなっていた垂直視差が，処理後では減少していることがわかる．理想的には画像の水平線上では垂直視差は存在しないはずである．しかし，図 4.3 左では，画像の水平線に当たる赤線上で視差が生じている．また，視差が最小になる部分が中心

より下にずれている．この様になった原因として，3.2 節で述べたカメラの固定ズレが考えられる．

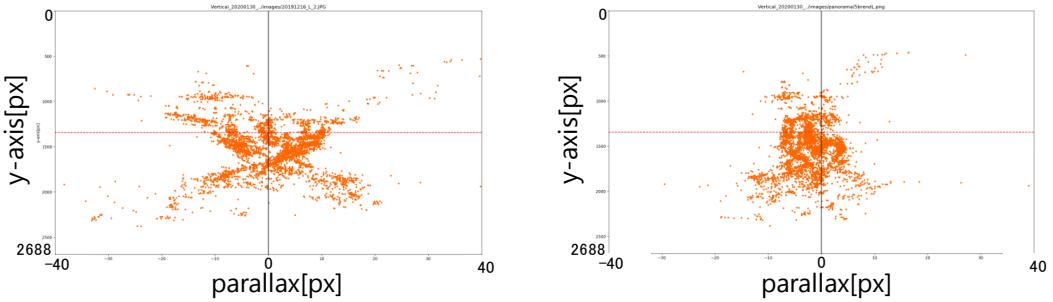


図 4.3　左：処理前の y 軸と垂直視差のグラフ　右：処理後の y 軸と垂直視差のグラフ

図 4.4 に処理前の視差の様子を，図 4.5 に処理後の視差の様子を示す．

図 4.4　処理前の視差の様子

図 4.5　処理後の視差の様子

図 4.5 の処理後の画像では，図 4.4 と比較して，画像全体に生じていた垂直視差が軽減されていることがわかる．特に，カメラの横方向や，画像の上下において，斜めだった視差の方向が水平に近くなっている．真横付近では，水平視差，垂直視差ともに小さくなっている．

処理前後の画像を HMD で観察して比較した．処理前の画像を観察すると，正面では立体に見えたが，横方向では物体が二重に見えた．これは，視差が縦方向に生じているためである．後ろ方向では視差が逆転しているため，物体が二重に見えた．

処理後の画像を観察すると，正面と後ろ方向で立体感が感じられた．横方向では，物が二重に見える事はなくなった．しかし，立体感は感じられなかった．横を向いたまま真上を見ると，蛍光灯が二重に見えた．これは，視差が縦方向に生じているためである．また，下を見ると，三脚が写り込んでいた．

4.2　理想的な視差との比較などの考察

視差の計算方法について考える．本研究では，2 枚の正距円筒画像で視差を評価した．正距円筒画像は，画像中央の水平線から上下に離れるにしたがって，シーンが横に引き伸ばされる．よって，画像の中央付近と比較して，上下部分では水平視差が増加する．そのため，本来は透視投影画像で比較するのが正しいと考えられる．

人間の視覚にとって理想的な視差は，水平方向だけの視差である．本研究で生成する立体 360° 画像には，ある程度の垂直視差が残るため，像が二重に見えるなど，立体視の障害になると考えられる．

本研究では，2 枚の正距円筒画像での視差が水平視差だけになることを一つの目標にしているが，そのような立体 360° 画像であっても，真上を見上げた場合などには，顔の向きと視差の方向が一致しない，という状態が生じる．

本研究の当初の目標は，全ての方向で立体視できる立体 360° 画像を生成することであった．しかし，横方向では立体視できないという課題が残った．この課題の解決も将来の課題である．

5 章 結論

本研究は全周囲で立体視可能な 360° 画像の作成を目標とした．まず，固定した 2 台の 360° カメラを用いて，2 枚の 360° 画像を撮影した．撮影した 2 枚の正距円筒画像から視差のグラフを作成し，そのグラフを観察して，行うべき処理を考えた．そして，お互いのカメラの写り込みを消去する，画像を入れ替えて対応点でつなぎ合わせる，色を合わせる，継ぎ目を滑らか変化させる，といった画像処理を施した．

その結果，カメラの前方向と後ろ方向では正しい視差になった．また，画像全体に生じている垂直視差を減少させることができた．これによって，HMD で観察したときに，物が二重に見える状態が軽減され，見え方が改善された．しかし，横方向において立体視できるようにはならなかった．

残った課題として，以下の 5 点がある．

① 横方向での立体視
当初の目標である全周囲で立体視できる画像にはならなかった．

② つなぎ合わせの自動化
画像の継ぎ目でズレを修正する際に特徴点を手作業で入力する必要がある．この課題の解決に，本研究室で過去に研究されたものが参考になると考える [2].

③ カメラの固定位置のズレ
撮影機材を手作業で調整しているため，撮影した 2 枚の画像は完全に揃っていない．画像のズレを修正するために，幾何学的な補正を行う必要がある．

④ 研究室以外の場所で試していない
本研究では，室内で撮影した画像のみを使用している．その他の場所でどのような結果になるか調べる必要がある．

⑤ 撮影機材が写り込んでいる
撮影した画像には治具，三脚や他方のカメラといった撮影機材が写り込んでいる．他方のカメラは消去したが，治具や三脚も鑑賞の妨げになるため，消去することが望ましい．

参考文献

[1] S. Peleg, M.Ben-Ezra and Y.Pritch, “Omnistereo: Panoramic Stereo Imaging,” IEEE Trans. on PAMI, Vol.23, pp. 279 - 290, 2001.

[2] 川西　洋嗣, “立体 360° 画像の自動合成に関する研究,” , 京都産業大学コンピュータ理工学部 特別研究 II, 2018.

[3] K. Matzen, M. F. Cohen, B. Evans, J. Kopf, R. Szeliski, “Low-cost 360 stereo photography and video capture,” ACM Trans. on Graphics, Article No 148, July 2017.

謝辞

本論文を作成にあたり,丁寧な御指導を賜りました蚊野浩教授に感謝いたします.

付録１　本研究で開発したプログラム

開発環境：Python 3.7.4，Visual Studio 2017，OpenCV4.1.0

[photo_change.py]

カメラの領域を，もう一方のカメラ画像で上書きするプログラム

[stitch4_one.cpp]

画像の後ろ方向を入れ替えて，特徴点が合うようにつなぎ合わせるプログラム

[min_pow.py]

最小二乗法を用いて係数を求め，画素値の濃淡変換をするプログラム

[image_brend.py]

画像の継ぎ目を，滑らかに変化させるプログラム

[netmatch0.py]

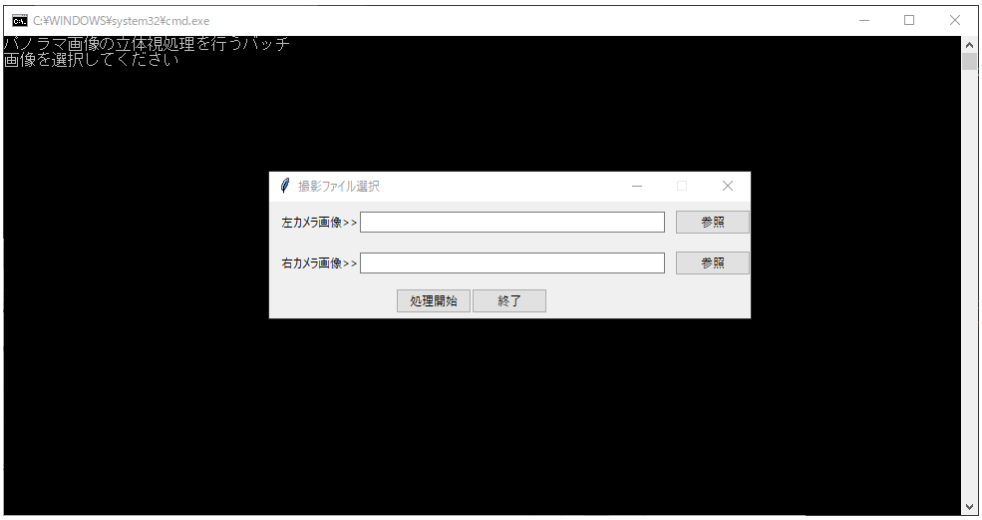
2 枚の画像から視差を計算して，グラフ化するプログラム

[exec.bat]

立体 360°画像の生成を実行するバッチファイル

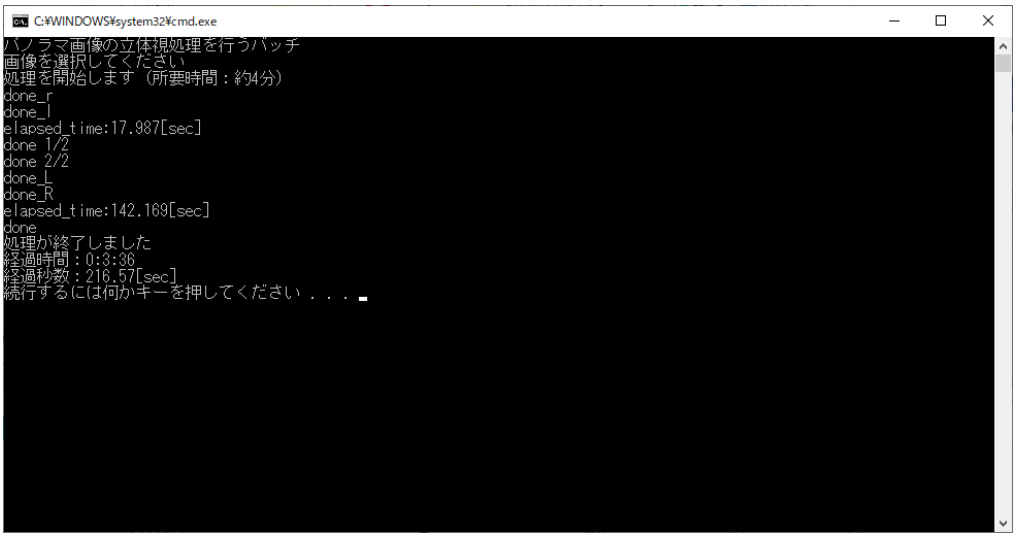
付録２　作成したプログラムの利用方法

バッチファイル exec.bat を実行すると画像を選択するウィンドウが表示される（図付録-1）.



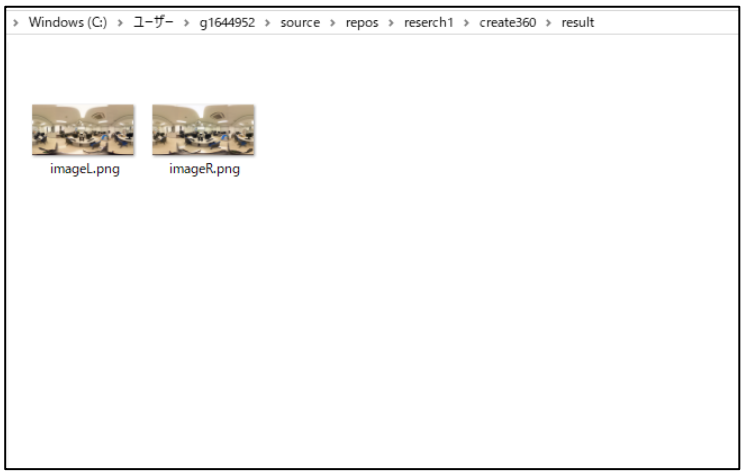
図付録-1　バッチファイルを実行した様子

参照ボタンを押して，左画像と右画像をそれぞれ選択する．2 つの画像を選択した後，処理開始ボタンを押下すると処理が開始される．使用した開発環境（CPU：インテル® Core™ i5-4590，メモリ：8GB，HDD：1TB）では約 4 分の処理時間がかかった．図付録-2 に終了後の出力内容を示す．



図付録-2　処理終了時のコマンドプロンプト

終了すると，result フォルダの中に処理された 2 枚の画像が保存されている（図付録-3）．temp フォルダには処理途中の画像が保存される．



図付録-3　実行後の result フォルダ

図付録-3 の画像を Unity に読み込ませて，HMD で鑑賞する．

このプログラムでは，つなぎ合わせの対応点を手動で入力しなければならない．対応点は，stitch5_path フォルダ内の Visual Studio ソリューションを開いて変更する．71,72,80,81,168,169,177,178 行目にそれぞれの対応点を入力する．コンパイルを行った後，create360¥stitch5_path¥x64¥Debug にある exe ファイルを，create360¥program フォルダにコピーする．