視線制御を用いた VR システムの一検討

池上貴博[†1]　尼崎太樹[†1]　久我守弘[†1]　飯田全広[†1]　末吉敏則[†1]

概要：筋ジストロフィー患者など身体を思い通りに動かすことができない場合を想定した，視線制御を用いた VR ゴーグルデバイスシステムを開発した．このシステムは VR(virtual reality)と視線制御を用いており，離れた場所の景色を見ることができる．

Construction of VR system using sight control

TAKAHIRO IKEGAMI[†1]　MOTOKI AMAGASAKI[†1]
MORIHIRO KUGA[†1]　MASAHIRO IIDA[†1]　TOSHINORI SUEYOSHI[†1]

Abstract: We developed a VR goggle device system using eye-gaze control assuming a case where muscular dystrophy patients and others can not move the body as desired. This system uses VR (virtual reality) and eye-gaze control, and you can see the scenery of a remote place.

1. はじめに

近年で視線入力装置に関する技術研究は盛んに行われており，眼球の動きのみで入力できる技術はパソコンのマウスポインタを操作する目的などで用いられている．視線入力を行うことによって得られるメリットは，マウスを手で操作してポインタを操作する作業を省略できる点や，マウスといった入力装置を利用することができない場合にも利用できるという点にある.また，現在筋萎縮性側索硬化症 Amyotrophic Interal sclerosis:ALS）という全身の筋肉へ命令を伝える運動神経細胞が侵される病気や，筋ジストロフィーという筋萎縮を起こす病気などがある．患者は身体を思い通りに動かすことができないケースが多く，眼球のみでしか他人とのコミュニケーションが取れない．こういった病気の患者や，何らかの要因で身体の自由が利かなくなり，マウスなどの入力装置が使えない場合には，視線入力装置の利用が有効である．本研究ではVRと視線制御を用いて，離れた場所の景色を視覚可能にする組込みシステムを構築した．構築したシステムは，まずステレオカメラという人の両目のようにレンズが搭載されたカメラから映像を取得し，サーバが動画として配信する．次に，その両目の動画をスマートフォンで受信しVRアプリケーションによって,立体的に体感できるように再生する.またこのとき，ユーザの視線情報を動画配信しているサーバに送信することで，ステレオカメラに取り付けたサーボモータを制御させ，カメラの向きを上下左右に変更することで視線変更を行う．この一連の流れが開発したシステムである．サーバには，低価格で小型のコンピュータボードである RaspberryPi3 を使用した.VRアプリケーションの開発には，Unityという統合開発開発環境を内蔵し複数のプラットフォームに対応するゲームエンジンを使用した．また，視線検出処理については自作のアルゴリズムとディープラーニングを用いた場合の2種類を作成した.構築した組込みシステムは遅延時間，また視線制御の正答率で評価する．遅延時間はステレオカメラの映像をVRアプリケーションが画面に再生するまでの実行時間と，視線検出からサーボ―モータを制御するまでにかかる実行時間を算出した．また，遅延時間に関しては人間が違和感なく認識できる30fpsを目標とする．

†1 熊本大学
Kumamoto University

2. システムの概要

実現したい AR システムは，ユーザーが VR ゴーグルを装着し，ゴーグル内に装着したスマートフォンによって遠くの映像を体感するというものである．また，単に遠くの映像を映し出すというものではなくユーザーが視線をゴーグル内で動かすことによりゴーグルで見る映像も連動して動くようにする．

ハードウェアとシステムの構成を図 1 に示す.本研究で作成するプロトタイプのシステムでは二つの RaspberryPi3(RaspberryPi 3 ModelB Single) を使用する．まず一つ目の RaspberryPi3 はユーザーが装着する VR ゴーグル(Zeiss 社製，VRONEPlus ヘッドセット) に取り付けてあり,ゴーグル内のユーザーの両目を RaspberryPi3 に接続したカメラモジュール(adafruit 社製，SpyCameraforRaspberryPi, メーカ番号:1937) で撮影する．このカメラモジュールはゴーグル内に設置する．撮影された両目の画像をもとに

ⓒ2016 Information Processing Society of Japan

RaspberryPi3 で視線検出処理を行う．処理の結果はユーザーから遠隔地にあるステレオカメラ側の RaspberryPi3 への送信を行う．また,VR ゴーグルには遠隔地にあるステレオカメラの映像を表示する Android 端末を装着してある．二つ目の RaspberryPi3 ではステレオカメラ側にある．こちらでは usb 接続しているステレオカメラ(SVPRO 社製，3DVRCamera)の映像を Android 端末へ送信し Unity アプリで映像を表示する．また，受信した視線検出の情報をもとに二つのサーボモータ(TOWERPRO 社製，MicroServo9g SG90)の制御を行いサーボモータの角度を変更する．図 2 にユーザーが装着するゴーグルの画像を示す．ゴーグルの内側の両目を撮影するカメラはレンズのサイズの都合上両目とは少し斜め上の位置に取り付けられ，目との距離は約 3.5cm である(図 3)．また，図 4 にステレオカメラ側の画像を示す．ステレオカメラは二つのサーボモータに固定され，モータは RaspberryPi3 の GPIO 端子に接続され制御される．

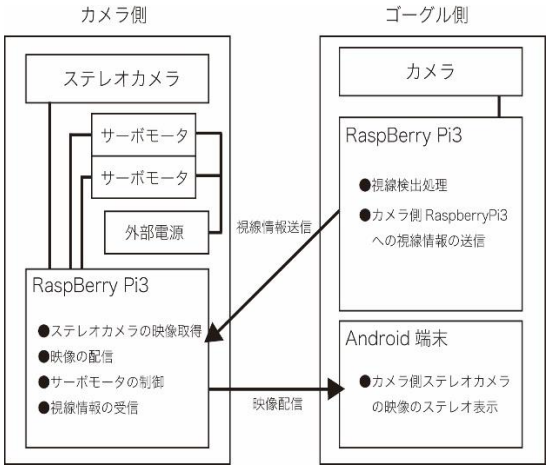

図 1　構成した組み込みシステムの概要
Figure 1　Overview of the built-in system that has been configured.

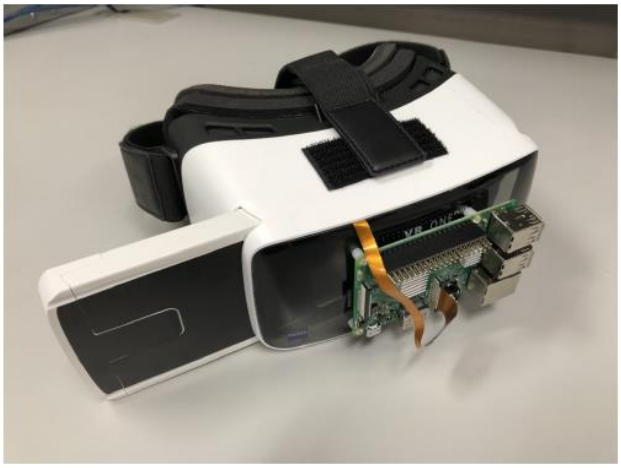

図 2　VR ゴーグル側の外観
Figure 2　Appearance of VR goggle side

図 2　VR ゴーグルと魚眼カメラ
Figure 2　VR goggles and fish eye camera

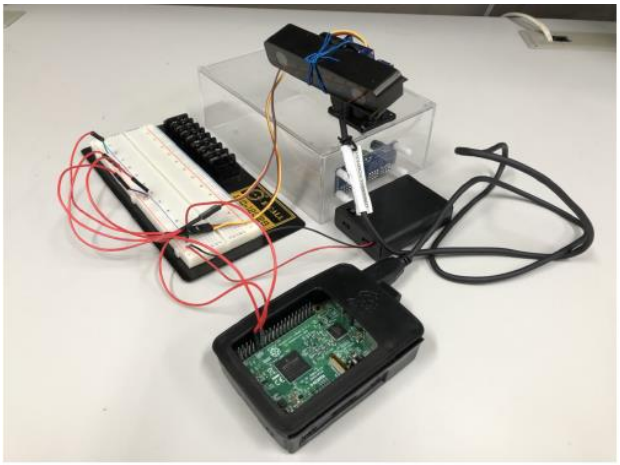

図 4　ステレオカメラ側の外観
Figure 4　Appearance of stereo camera side

3. 提案手法

今回システムを作成する際に用いた動画配信ソフト MJPG-streamer，今回作成した視線検出アルゴリズム，またディープラーニングの視線検出について説明する．

3.1 MJPG-streamer

MJPG-streamer とは RaspberryPi3 に接続したカメラから映像を取得し配信することができる動画ストリーミングソフトウェアである．このソフトウェアは，解像度やフレームレート，配信するポート番号など多様な設定ができる．

3.2 重心座標を用いた視線検出

二値化した画像の重心の位置を利用する視線検出手法を作成した.まず魚眼レンズのキャリブレーションを行い，歪んだ両目の画像の校正を行う．両目の画像を適切な閾値

ⓒ2016 Information Processing Society of Japan

によって二値化処理し，白目と黒目の部分を明確に分類した二値化画像を準備する．次に二値化した画像の両方の目の領域をあらかじめ指定し，左右両目の領域のトリミングをそれぞれ行う．これは視線検出に不必要な目以外の部分を取り除くとともに，画像のサイズを小さくすることで処理の負担を減らす目的がある.1280×1024px の画像に対し，右目を y 軸 500px から 750px と x 軸 100px から 400px の範囲でトリミングし,左目を y 軸 500 から 750px と x 軸 800px から 1100px の範囲でトリミングする．

その後ゴーグル内に映し出された映像の中心を見たときの左右の目それぞれの二値化画像の重心座標を取得しその座標の値を保存する．ゴーグル内のカメラモジュールからキャプチャーした画像に対して逐一二値化処理と重心座標の計算を行い，重心座標の変位量によって視線方向を検出するという仕組みである．これは，瞳の位置が見る方向に連動して動く特性を利用したもので，両目の瞳の領域の重心と実際の瞳の中心はある程度一致するという想定のものである.また，今回はサーボモータを上下左右に動かしたいので，視線の動きを上下左右の 4 方向の検出ができればよい．なお視線検出処理では画像処理を多用するため，本研究ではプログラミング言語 Python を用い，画像処理用のライブラリ OpenCV を使用する．

変化の閾値の設定瞳の動きに伴う二値化画像の重心座標の変化で視線方向を検出するためには，上下左右への重心位置の変位量のサンプルを取る必要がある．ゴーグルに差し込んだスマートフォンに 5 つのポイントを配置し,画像の指定点を見たときの重心座標のサンプルを取る．左右の目それぞれの二値化画像に対し重心座標を算出し，重心の x 座標 y 座標の変位量が設定した閾値よりも変位した場合に上下左右それぞれを向いているものとする.閾値は,x 軸方向に 17 移動した場合を右,-17 移動した場合を左,y 軸方向に 10 移動した場合を下,-10 移動した場合を上と設定した．

3.3 ディープラーニング

ディープラーニング（深層学習）とは，多層のニューラルネットワークによる機械学習の手法のことであり，十分なデータ量があれば人間の力なしに機械が自動的にデータから特徴を抽出するという点で優れている. これは音声認識，画像認識，自然言語処理といった問題に対して高い性能を示しており，様々な分野で活躍している．今回は，この機械学習において AlexNet*1 という CNN を用いた. また学習用データとして中，右，左，上，下の 5 方向を向いた写真を 18 人分撮影を行った．この時，中方向の画像は他の方向に比べ認識しにくいと判断し，中方向の画像は 15 枚，それ以外は 5 枚ずつ撮影した．また，この画像を学習用データと検証用データに分けた後，DataAugmentation という手法を用い学習用のデータを 10,500 枚，検証用データを 2,100 枚まで増幅させた．DataAugmentation には，ガウシアンノイズ, ごま塩ノイズ, ランダムに回転, ランダムにシフト, ランダムにズーム，チャンネルをランダムに変更させるなどといった処理を行った．処理前の画像を図 4，DataAugmentation を行った後の画像を図 5 に示す.

図 4　目の加工前の画像
Figure 4　Image before eye processing

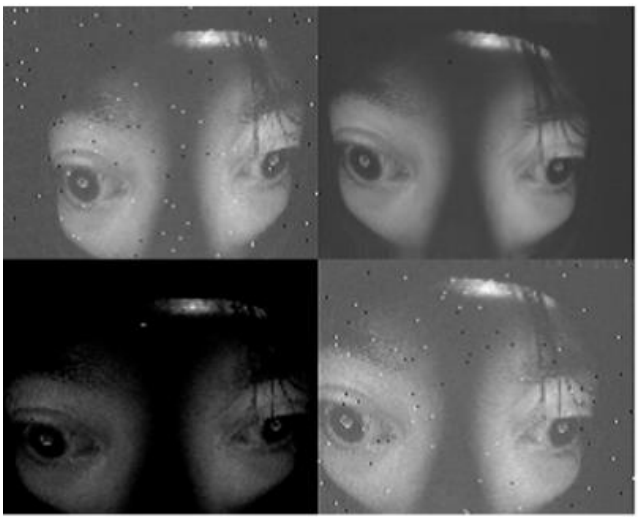

図 5　目の加工後の画像
Figure 5　Image after eye processing

4. 評価

評価評価項目は以下の 2 項目である
・遅延時間
・視線の認識率

遅延時間は VR と視線制御が高いリアルタイム性を必要とするため評価する．また，全体の遅延時間を算出し，一般

ⓒ2016 Information Processing Society of Japan

的に違和感なく認識できるという 30fps を達成できるかどうか調査する．

4.1 評価環境

表 1 に，評価を行った際の環境について書く．作成したプログラムは RaspberryPi3 上で動かすものとする．また，RaspberryPi3 に載っている ARM は Cortex-A53(1.2GHz)である．

4.2 遅延時間

遅延時間は VR 処理と視線制御処理の 2 種類算出する．VR 処理はステレオカメラの映像を RaspberryPi3 が配信して VR アプリケーションがスマートフォンに再生するまでの処理とする．また視線制御処理は VR ゴーグルの RaspberryPi3 が視線を検出しもう一方の RaspberryPi3 に情報を送信してサーボモータに制御信号を送るまでの処理とする．VR 処理における，RaspberryPi3 が MJPG-streamer で配信を開始するのにかかる時間を T1[ms] とし，スマートフォンが RaspberryPi3 から動画を受信し VR 表示するまでの時間を T2[ms] とする．また視線制御処理における，魚眼カメラによりユーザの視線を撮影して画像処理を行い重心座標を用いた視線検出を行うまでの実行時間を T3[ms] ，機械学習を用いた視線検出の実行時間を T’3[ms] とし，視線情報をもう一方の RaspberryPi3 間で往復させるまでの時間を T4[ms], 受信した視線情報をもとにサーボモータに制御信号を送るまでの時間を T5[ms] とする．計測結果を表 1 として示す.

表 1　遅延時間

Table 1　Delay time.

T1[ms]	T2[ms]	T3[ms]	T3’[ms]	T4[ms]	T5[ms]
5.24	17.78	830	673	0.35	0.05

4.3 認識精度

重心座標を用いた視線認識法と，機械学習を用いた手法において，視線認識の精度を評価した．計測結果を表 2 に表す．また，重心座標を用いた視線認識を A，機械学習を用いた視線認識は B とする．

表 2　認識精度

Table 2　Recognition accuracy.

	中	右	左	上	下	Ave
A	56%	60%	62%	78%	73%	66%
B	94%	99%	99%	99%	92%	97%

5. 考察

視線制御に関する 2 つの手法について，実行時間 と認識精 度を比較した．実行時間に関しては，機械学習 を用いた視線制御手法が重心座標を用いた視線制御の手法よりも 157[ms]短かった．しかし，この数値は目標とする 30fps にほど遠く，この視線制御部をいかに高速化するかが今後の課題となる．また，認識精度に関しては機械学習を用いた視線制御手法が圧倒的に高かった．重心座標を用いるよりも，機械学習を行い CNN を用いた方がより高い精度を得られることが分かった．これらの結果から，処理速度の解決法として FPGA(field-programmable gate array)を用いることを検討している．FPGA は並列処理を行うことが可能であり，同じような処理を複数回行う画像処理とは相性がいいと言える．

6. まとめ

今回は，筋ジストロフィー患者など身体を思い通りに動かせない場合を想定した，視線制御を用いた VR デバイスシステムを開発した．またこのシステムを開発する際に，VR 酔いをしないためにも 30fps を出す必要があった．そのためにはシステム全体における遅延時間が 1/30[s]を切らなければならなかった．今回視線制御の視線を検出する手法に関して，重心座標を用いた視線検出法とディープラーニングを用いた視線検出法の 2 つの種類を検証した．しかし，このどちらも実行時間が 30fps には満たしていなかった．この結果から，このシステム全体を高速化するためには視線制御処理部に関してより工夫する必要がある．

7. 参考文献

[1] Alex Krizhevsky, ImageNet Classification with Deep Convolutional Neural Networks,
[2] 巣籠悠輔,詳解ディープラーニング　TensorFlow・Keras による時系列データ処理
[3] 宇田周平,林宜憲著,Raspberry Pi2 と Windows10 ではじめる IoT プログラミング,株式会社マイナビ,2015.
[4] 岩井雅幸著,uGUI ではじめる Unity UI デザインの教科書,株式会社マイナビ,2015.
[5] Jonathan Linowes 著,高橋憲一,安藤幸央,江川崇,あんどうやすし訳,Unity による VR アプリケーション開発,株式会社オライリー・ジャパン,2016.
[6]斉藤康毅,ゼロから作る Deep Learning -Python で学ぶディープラーニングの理論と実装,株式会社オライリー・ジャパン,2016.

ⓒ2016 Information Processing Society of Japan