

2022 年度（令和 4 年度）

創造工学セミナー II Final Report

非接触型ハンドジェスチャ
インタフェースの開発

チームメンバー

S5-19016 國吉 諒太

S5-19051 溝口英悟

S5-19060 吉岡 嵩太

指導教員：金丸 隆志 教授

目次

第 1 章 緒言(溝口担当)	1
1.1 研究背景	1
1.1.1 新型コロナウイルスの蔓延	1
1.1.2 非接触操作の需要増加	1
1.2 類似研究	3
1.2.1 任意の位置での簡単なジェスチャによる家電操作システム	3
1.2.2 自然な手振りによる直感的なハンドジェスチャ UI	4
1.2.3 類似研究の問題点	5
1.3 研究テーマ	6
1.3.1 研究初期のコンセプト及びテーマ決定までの軌跡	6
1.3.2 非接触式路線バス降車ボタン(Touchless Stop Button)	7
1.3.3 本研究の目標	8
第 2 章 システムの概要(吉岡担当)	9
2.1 想定環境	9
2.1.1 TSB を利用できるバス	9
2.1.2 バス内の測定	10
2.2 ハードウェア	13
2.2.1 検出用カメラ	13
2.2.2 システム動作用パソコン	16
2.3 ソフトウェア	17
2.3.1 開発環境及びライブラリ	17
2.4 MediaPipe	18
2.5 測定値を基にしたカメラ位置の決定	19
第 3 章 本システムに用いる認識とポーズ判定(吉岡担当)	22
3.1 手の骨格検出	22
3.2 ポーズ判定	23
3.2.1 ポーズ学習	24
3.2.2 学習の手順	24
3.2.3 研究初期のポーズ	25
3.2.4 問題点	26

第4章 本システム実用化のための工夫(國吉、溝口担当)	27
4.1 複数カメラへの対応	27
4.2 2段階化	28
4.3 認識結果の処理	29
4.3.1 最頻値ポーズの導入	29
4.3.2 収集データ数の削減	30
4.3.3 移動最頻ポーズの取得	30
4.4 ユーザーによる処理時間の差異	32
4.5 複数個の手の認識	34
4.6 複数人の手の骨格検出	36
4.6.1 複数人の手の検出の検証	36
4.6.2 Frames Per Second 計測	37
4.7 誤認識の低減	39
4.7.1 問題点	39
4.7.2 誤認識防止ポーズの追加	39
4.7.3 検証及び結果	39
4.8 ポーズの改善	41
4.8.1 Natural User Interface	41
4.8.2 アンケート及び結果	41
4.9 フィードバック	43
4.9.1 バス環境の再現	43
4.9.2 実装したフィードバック	47
4.9.3 開発者画面の動作内容	48
第5章 TSB の評価(國吉担当)	52
5.1 評価方法	52
5.2 結果	53
5.3 課題	53
第6章 結論(國吉担当)	54
参考文献・URL	55
謝辞	58

第 1 章 緒言(溝口担当)

1.1 研究背景

1.1.1 新型コロナウイルスの蔓延

2019 年末、中国武漢市で発見された新型コロナウイルスは、その高い感染力から世界中でパンデミックを引き起こした。日本においても、2020 年 1 月に国内第 1 例目の感染者が検知されてから現在に至るまで、コロナウイルスは姿形を変え感染者を増やし続けている[1-1]。実際に、厚生労働省が公表している累積陽性者数のデータ[1-2]によると、2022 年 1 月 1 日時点での累積陽性者数は、1,728,655 人であるのに対し、2023 年 1 月 1 日時点では 29,299,459 人と、この 1 年間で 27,570,804 人も陽性者数が増加したことが分かる。データからわかる通り、今もなおコロナウイルスの感染拡大は留まることを知らない。これからも、感染症に罹患するリスクから身を守るために、密閉・密集・密接の三密を避けることが求められるだろう。

1.1.2 非接触操作の需要増加

社会全体で身体的距離の確保が求められる中、個人では人と物の距離の認識が変化しつつある。新型コロナウイルス(SARS-CoV-2)の感染経路は主に 3 つあり、空中に浮遊するウイルスを含むエアロゾルを吸い込むこと(エアロゾル感染)、ウイルスを含む飛沫が露出した粘膜に付着すること(飛沫感染)、ウイルスを含む飛沫を直接又は間接的に触った指で露出した粘膜を触ること(接触感染)である[1-3]。この 3 つの内、エアロゾル感染と飛沫感染は身体的距離の確保を意識することで対処が可能である一方、接触感染は無意識のうちに感染源に触れてしまうため対処が難しい。そのため多くの人々は、不特定多数が触れるものに触れないよう意識をするようになった。実際に、2020 年 5 月に実施されたコロナショック前後のモノとの接触に関する意識調査[1-4]では、79.8%がコロナ過前と比較して物との接触が気になるようになったと回答している。また、お店や施設などでやってほしい対策というアンケート項目では、過半数の 53.6%がパネルやボタンなどを触らないで操作できるようにしてほしいと回答している。そこで近年、人々の非接触ニーズを意識した非接触操作が可能な製品が多く誕生した。例えば、国内大手航空会社では空港内の自動チェックイン機(図 1)、自動手荷物預け機のタッチパネルが接触化されている(図 2)。このように非接触機器のニーズが高く、実際に接触機器の非接触化が始まっていることから、非接触操作ユーザーインターフェースを研究テーマとした。その中で、公共性の高い分野に我々はテーマの焦点を絞り、路線バスの降車ボタンを非接触化するシステムの開発を目指すことが決定した。



図1 自動チェックイン機[1-5]



図2 自動手荷物預け機[1-5]

1.2 類似研究

本節では、研究に関連する類似研究とその問題点を示す。

1.2.1 任意の位置での簡単なジェスチャによる家電操作システム

顔らは、簡単なジェスチャで家電を操作できるシステムを開発した[1-6, 1-7]。このシステムは、複数のカメラ画像から人の手振りを検出し、その位置で擬似相対座標系(図 3)を展開することで、部屋中任意の場所での家電操作を可能にしている。このシステムの欠点は処理時間がかかること、操作する際に何度も手を振る必要があり、使用者が疲労することである。この問題に対し、顔らは、手を1箇所に置いて短時間静止する手かざしジェスチャを利用することで使用者の負担を減らしている[1-7]。システムの操作精度と時間の評価実験を行った結果、8割以上の認識率が得られ、従来手法と比べ2倍以上高速化していることが示されている。しかしながら、家電操作後に使用者自らが定義した擬似相対座標系を消去する手順が必要であり、ユーザビリティに悪影響を与えていると我々は考えた。

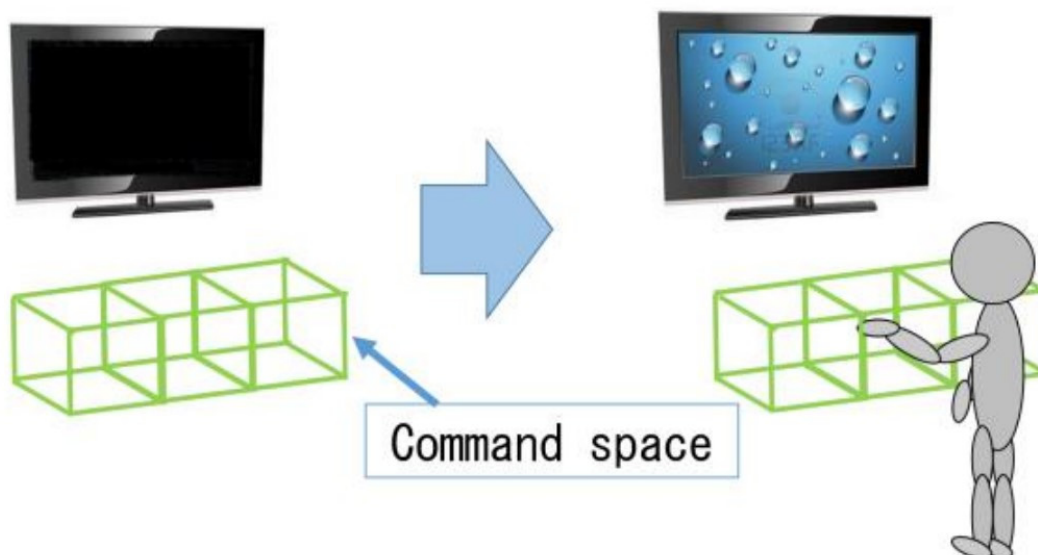


図 3 相対座標系展開のイメージ[1-7]

1.2.2 自然な手振りによる直感的なハンドジェスチャ UI

中州らは、手振りによる直観的なハンドジェスチャ UIを開発した[1-8]。ハンドジェスチャ UIでは、多くの機能を実現できること、ユーザーが簡単に、負担を感じずに操作できることが重要である。しかしながら、システムの機能を増やすと、それに対応したジェスチャの種類を増やさなければならず、ユーザーの負担が大きくなる。そのため、この研究では、システムの操作を少ない種類のジェスチャで可能にし、ユーザーへの負担を減らすことを目標としている。使用したハンドジェスチャ操作は主に3つあり、それらの操作方法を図4に示す。1つ目は、上下左右の手払い動作によるカーソルと画面の移動である。2つ目は、手のひらを握るといった手の形状による操作開始と決定である。3つ目は、手振り動作によるキャンセルの操作である。以上の3つのジェスチャにより、ユーザーへの負担軽減を可能にしている。また、評価実験の結果、手振り操作の認識成功率は96.3%であることが示されている。しかしながら、ユーザーによって実験結果が大きく変化しており、認識方式に工夫が必要であると我々は考えた。

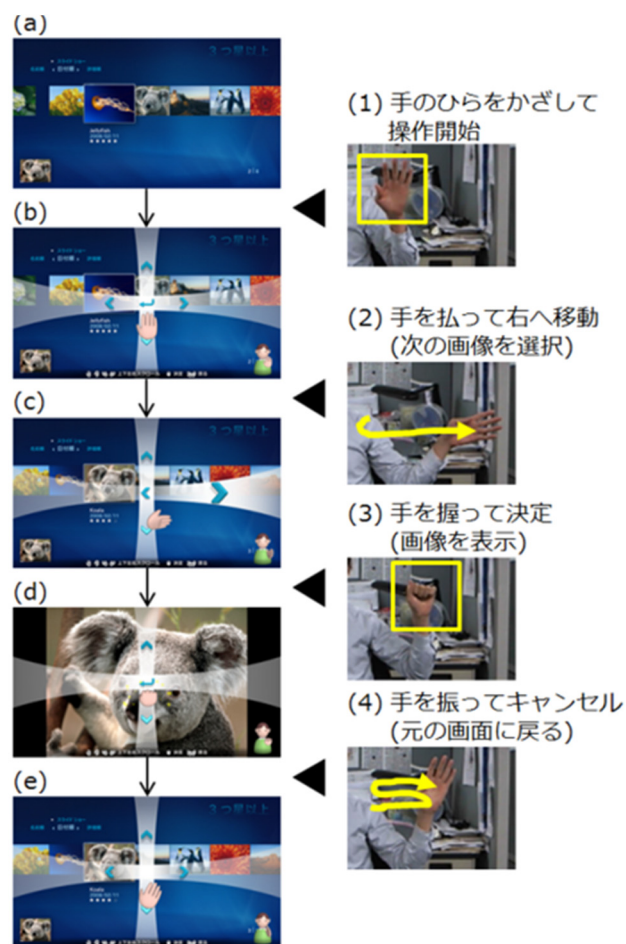


図4 中州らが開発したハンドジェスチャ UI[1-8]

1.2.3 類似研究の問題点

先に挙げた類似研究には下記のような問題点があると我々は考えた。

- A) 顔らのシステムでは、使用者自らコマンド空間を定義し、操作終了後に消去する必要がある。
- B) 顔らのシステムでは、使用者からコマンド空間までの距離が 0.1m から 0.9m と幅広く (図 5)、コマンド空間の場所を頭の中で思い描きながら操作する必要がある。
- C) 中州らのシステムでは、使用者の手振りの大きさ、速さ、移動距離で認識率が変動する。

上記で挙げた問題点のうち A と B はユーザビリティが大きく損なわれてしまう問題であり、C はインタフェースとして致命的な問題であるため大きく改善する必要があると思われる。

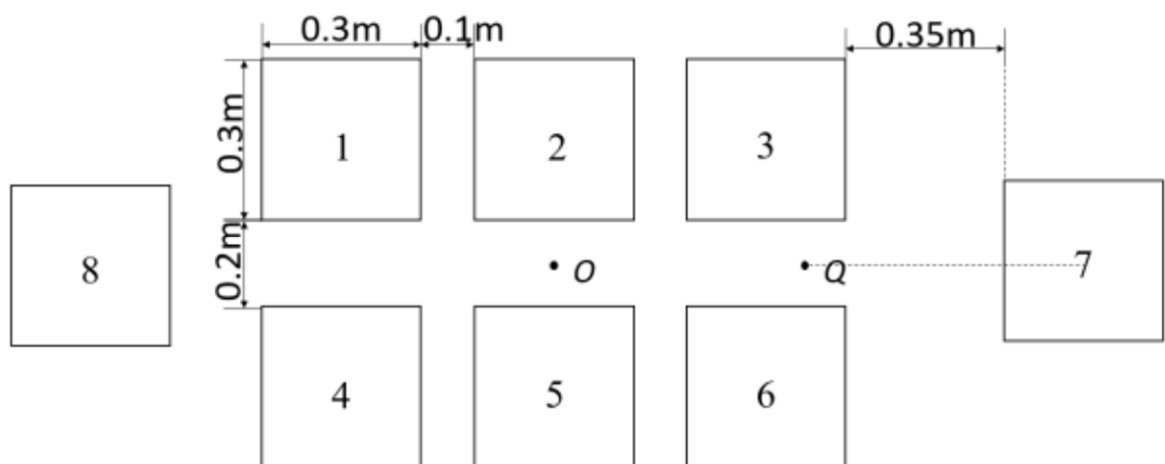


図 5 操作空間の距離[1-7]

1.3 研究テーマ

本節では、本研究のテーマについて述べる。

1.3.1 研究初期のコンセプト及びテーマ決定までの軌跡

テーマ選考をするにあたって「人の役に立つもの」「日々の生活を豊かにするもの」を初期コンセプトとしていた。さらに、我々は学生時代の大半を新型コロナウイルスが流行する中で過ごした。そこで、感染症対策として注目されていた「非接触化」を主軸として研究を行うことに決定した。その具体的な構想として、図6のような観光案内掲示板の検討を行った。その後、準備を進める中で2つの問題点が明らかになった。1つ目は、図6の観光案内掲示板は専用の端末での動作を想定しており、導入業者にとってのハードルが高いことである。2つ目は、より汎用性の高い非接触操作システムの開発を目指していることである。観光案内掲示板は代用の利かない独特な動作が含まれており、汎用的な非接触操作ではない。そこで、路線バス降車ボタン(図7)に着目した。ボタンを押すという動作には汎用性があり、我々の目標に合致していると考えたためである。具体的な説明は次節で行う。



図6 観光案内掲示板[1-9]



図7 バス降車ボタン[1-10]

1.3.2 非接触式路線バス降車ボタン(Touchless Stop Button)

従来の路線バスでは、整理券発行機や運賃箱といった様々な機器によって運行が管理されている。我々は、それらバス内機器の中で不特定多数の人々が触れる路線バス降車ボタンに着目し、非接触化を目指す。Touchless Stop Button(TSB)の概要を図8に、フローチャートを図9に示す。ユーザーの手のポーズ動作により、降車の合図を送るシステムである。まず、カメラの撮影範囲内にて、ユーザーは降車するためのポーズ動作を行う。次に、カメラによって取得されたポーズ動作をプログラムにより認識する。最後に、ポーズ動作が適切なものであると認識されれば、降車の合図が成立したとする。また、この間、ユーザーにとっての快適な降車を補助するために、音声案内も適宜行う。

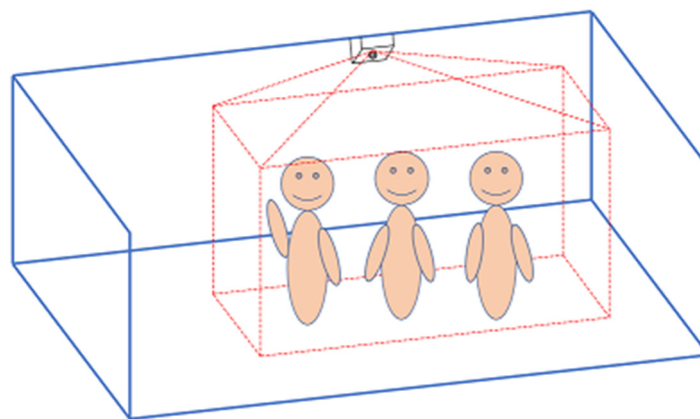


図8 TSBの概要

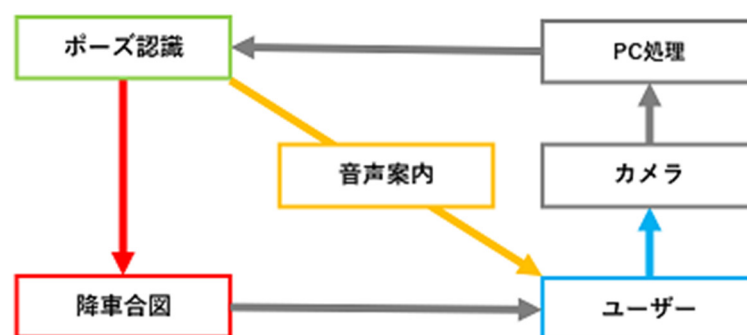


図9 TSBのフローチャート

1.3.3 本研究の目標

本研究の目標は下記の通りである。

- バスの環境を再現した状況で同時に 10 人が利用できるシステムとすること。
- 同じ状況で「バス降車」の意思表示の成功率が 95%以上であること。

さらに、到達目標を達成するために以下の小目標も設定する。

1. 使用者の位置にとらわれないシステムであること。
2. 認識に最適なカメラ角度と位置を決定すること。
3. 誤認識を低減させること。
4. 認識の信頼性を高めること。
5. 使用者への負担を減らすこと。
6. 使用者によって結果が左右されないこと。
7. 個々の手に応じた結果が出力されること。
8. 使用者へのフィードバックを用意すること。

第2章 システムの概要(吉岡担当)

2.1 想定環境

2.1.1 TSB を利用できるバス

TSB を利用するバスとして考えられるのは下記4種である。

図 10 路線バス	
図 11 高速バス [2-1]	
図 12 マイクロバス [2-2]	
図 13 乗用車型送迎者 [2-3]	

以上の候補から、路線バスを想定環境として用いることにする。路線バスは4つの中で最も降車ボタンの使用頻度が高く、本テーマとの親和性が高いためである。

2.1.2 バス内の測定

本研究で開発する TSB は、バス乗車時の利用を想定しているため、実際のバス車内での環境を再現する必要がある。そこで、西東京バス株式会社様のご協力の元、バス車内の寸法測定を行った。図 14 に、測定を行ったバスの外観、図 15 に実際に測定をしている様子、そして、測定した寸法を表 1 に示す。また、測定された寸法を追記した平面イメージを図 16 に、側面イメージを図 17 に示す。



図 14 測定を行ったバスの外観



図 15 測定の様子

表 1 バス車内の寸法

全長	8.2m
幅	2.07m
高さ（車内前方）	2.28m
高さ（車内後方）	1.86m

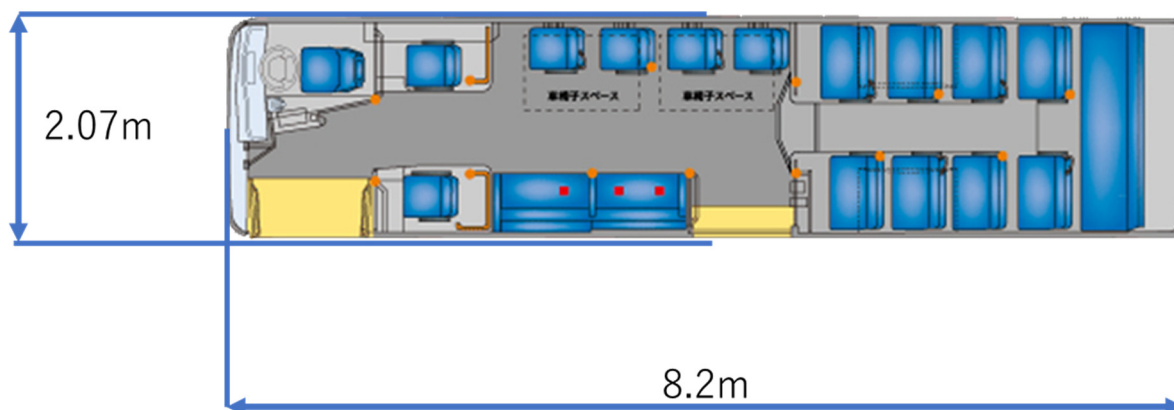


図 16 車内平面イメージ[2-4]

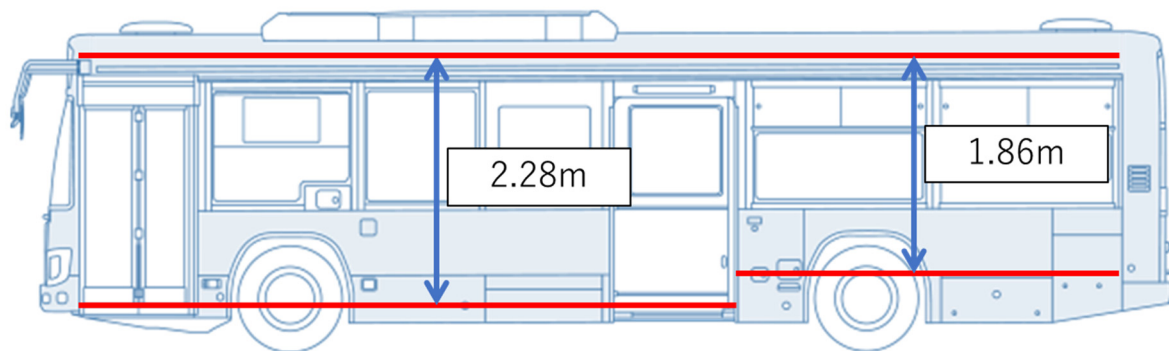


図 17 車内側面イメージ[2-5]

2.2 ハードウェア

本節では、研究で用いるハードウェアの説明を行う。

2.2.1 検出用カメラ

本研究で用いる検出用カメラの検討を行った。

まず検討したのは図 18 の 360° カメラである。このカメラで撮影した画像データに手の骨格検出をさせて簡単な動作確認を行う。なお、手の骨格検出の詳細は 3.1 で述べる。



図 18 Ricoh theta(360 度カメラ)

その結果、図 19 のようにそのままの状態では手が十分に画面内に写っていても手の検出に成功しなかった。



図 19 広角の画像(theta にて撮影)

画像を一般的な縦横比率(4:3)にトリミングし、改めて試したところ図 20 のように手が検出された。



図 20 広角画像を 4:3 にトリミングしたもの

このことから、広角の画像では手の骨格検出が難しいという結論となった。解決策として、広角の画像に図 21 のイメージ画像のように、1 枚の画像に対して位置を変えて複数回認識を行うことが挙げられるが、それでは、複数のカメラに対して検出を行うことと変わらない。そのため、複数カメラを対象に研究を行うという結論になった。一台の広角カメラよりも複数のカメラを用いた場合の方が多くの範囲、乗客を検出出来るという理由もある。

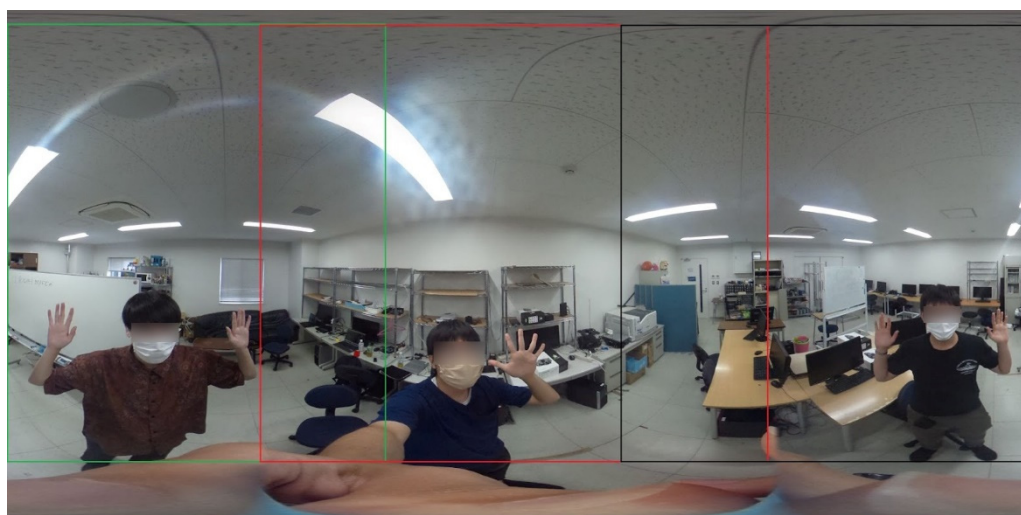


図 21 広角画像に対する複数回の検出のイメージ

図 22 および表 2 は本研究で用いることになった web カメラの詳細である。



図 22 使用する web カメラ[2-6]

表 2 web カメラ仕様[2-6]

型番	Logitech C920 n
最大解像度	1080 p/30 fps - 720p/ 30 fps
カメラ画素数 (メガピクセル)	3MP
フォーカスタイプ	オートフォーカス
対角視野 (dFoV)	78°
寸法 (高さ×幅×奥行)	43.3 mm×94 mm×71 mm
重量	162g
対応 OS	Windows 8 以降 macOS 10.10 以降

2.2.2 システム動作用パソコン

システム動作用パソコンとして HP 社製の HP Spectre x360-15-df0009tx(図 23)を用いる。
詳しい仕様を表 3 に示す。



図 23 HP Spectre x360-15-df0009tx[2-7]

表 3 パソコンの仕様[2-7]

型番	HP Spectre x360-15-df0009tx
OS	Windows 10 Pro(64bit)
CPU	インテル® Core™ i7-8750H プロセッサー (2.20GHz-4.10GHz, インテル® スマートキャッシュ 9MB)
GPU	NVIDIA® GeForce® GTX 1050Ti with Max-Q Design グラフィックス
メモリ	16GB

2.3 ソフトウェア

2.3.1 開発環境及びライブラリ

システムの開発にあたり、機械学習やデータサイエンスに必要なライブラリ等を Anaconda によりインストールした。開発環境は Anaconda に含まれる Spyder を利用し、言語は Python 3.7.13 を用いた。

使用したライブラリ等を表 4 に示す。

表 4 使用したライブラリ等

名称	バージョン
MediaPipe	0.8.1
OpenCV	3.4.2
Tensorflow	2.3.0
Numpy	1.12.5
Playsound	1.2.2

2.4 MediaPipe

MediaPipe[2-8]とは Google 社が提供するライブストリーミングのためのオープンソースのML(機械学習)ソリューションである。MediaPipe は Android、iOS、C++、Python、JavaScript などの言語で ML プラグインを構築できる。本研究では Python により MediaPipe を用いる。MediaPipe には画像や動画から得たデータを元に様々な種類の推論を行うソリューションが図 24 のように用意されている。

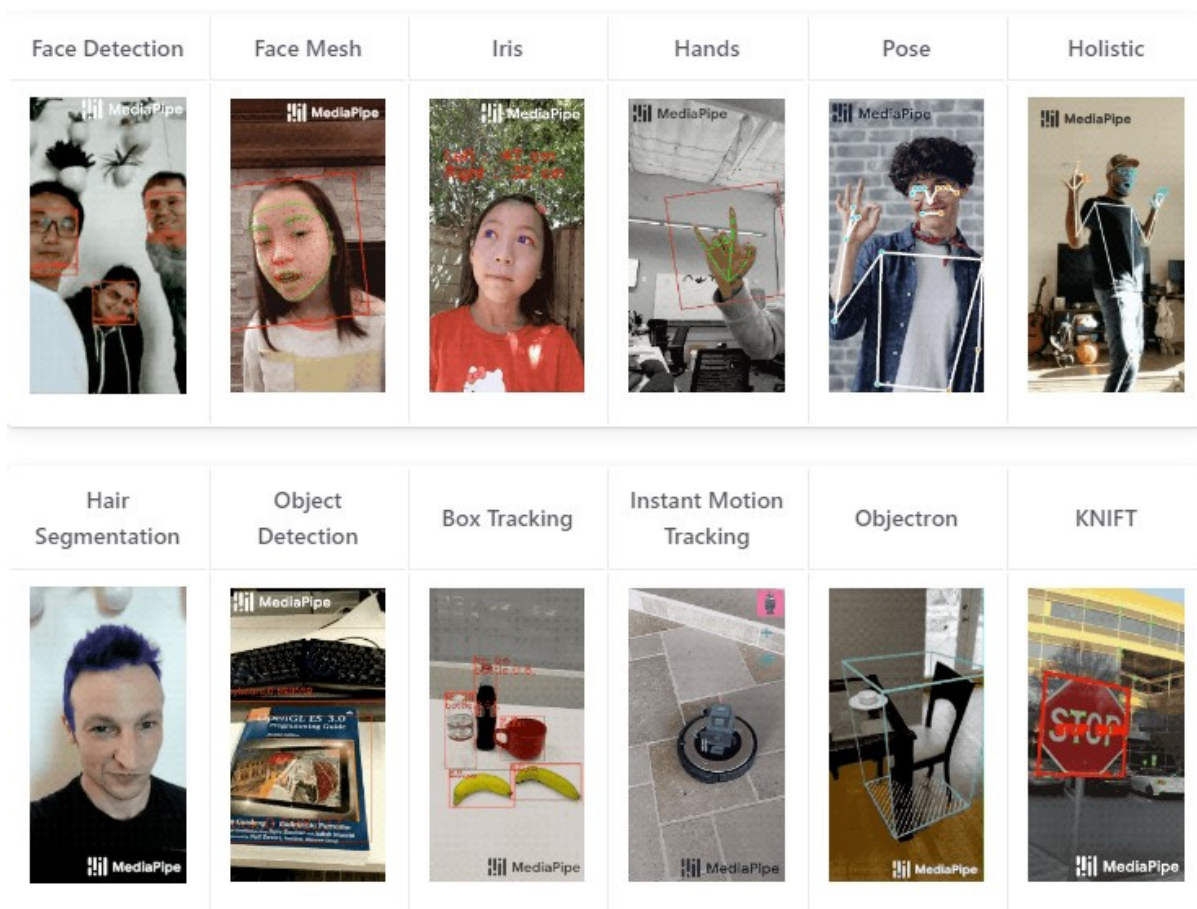


図 24 MediaPipe[2-8]

本研究では、このうち Hands による手の骨格検出の機能のみを用いて TSB を実現する。

2.5 測定値を基にしたカメラ位置の決定

検出に適したカメラの位置を決定するため、2つの検証を行った。1つ目は、距離が検出精度に与える影響を確認する検証である。この検証では MediaPipe を用いて、手の骨格描写が維持される距離を調査した。検証は下記の手順で行った。

1. バス内の高さ 2.2m の位置にカメラを設置する。
2. カメラからの距離が 0.5m、1m、1.5m、2m、2.5m、3m、3.5m の位置に立つ(図 31)。
3. 手をカメラに向ける。
4. 骨格が描写されるか否かを確認する。
5. 手を後ろで組み、骨格描写がされていない状態に戻した上で 3~4 の流れを各位置で 100 回ずつ行う。

検証結果は表 5 の通りである。

表 5 検証結果

距離	0.5m, 1m, 2m, 2.5m	3m	3.5m 以上
検出率	100%	94%	0%

実験の結果 3.5m より遠い距離において検出が行われなかった。この結果から、手の骨格検出には一定以上の大きさの手が必要であると考えられる。

次に、カメラの設置角度が検出精度に与える影響を調べる。下記の手順で行った。

1. バス内の高さ 2.2m の位置にカメラを設置する。
2. 0.5m、1m、1.5m、2m、2.5m の位置に立つ(図 31)。
3. 30°、45°、60°、75°、90° にカメラを傾ける。
4. 手をカメラに向ける。
5. 骨格が描写されるか否かを確認する。
6. 手を後ろで組み、骨格描写がされていない状態に戻した上で 4~6 の流れを各位置、各角度で 100 回ずつ行う。

検証結果は下記の通りである。

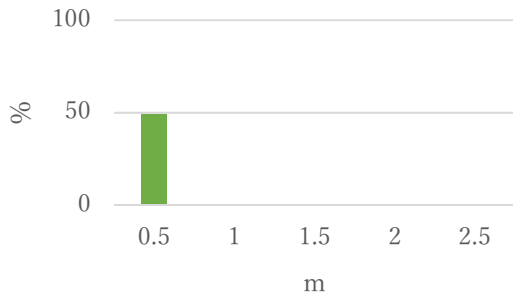


図 25 30°

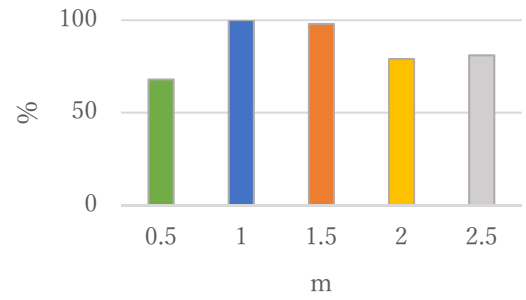


図 26 75°

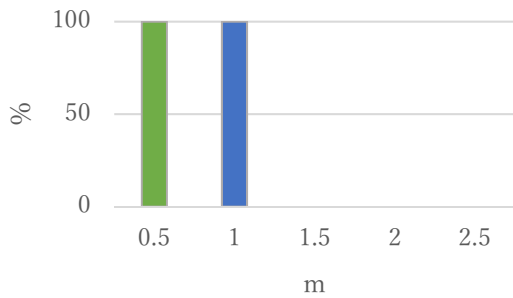


図 27 45°

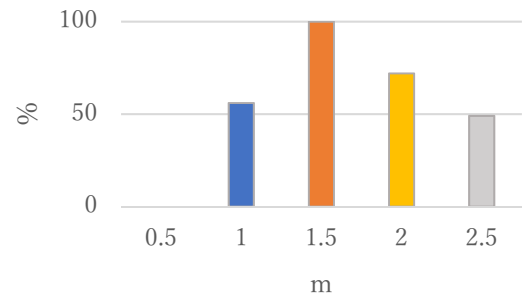


図 28 90°

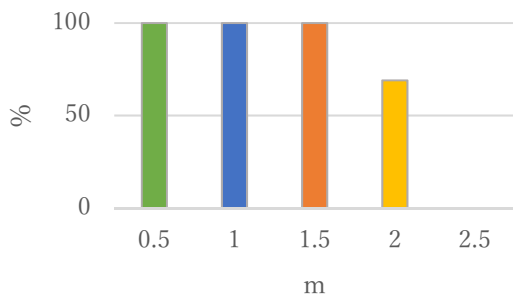


図 29 60°

検証結果から、角度別の検出可能範囲は 75° が最も高いことが判明した。しかし本研究では、検出の信頼性を重要視しているため、0.5~1.5m における検出の信頼性が高く 2 番目に検出範囲が広い 60° を採用した。

以上 2 つの検証から、検出に適したカメラの距離は 0.5~1.5m でカメラの角度は 60° と判明した。これらを測定したバスの寸法に合わせて、バス全長 8.2m に対して 1.5m 間隔で 60° の位置にカメラを設置し、範囲内を全て検出するためにはカメラが 5 台必要になると判明した(図 30)。

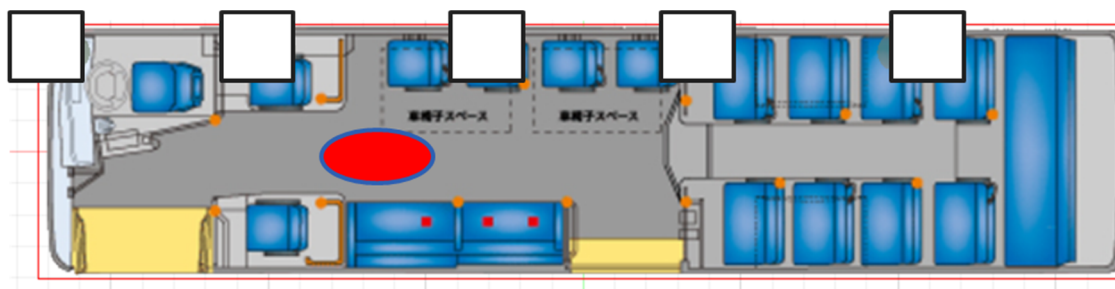


図 30 カメラを設置すべき位置[2-4]

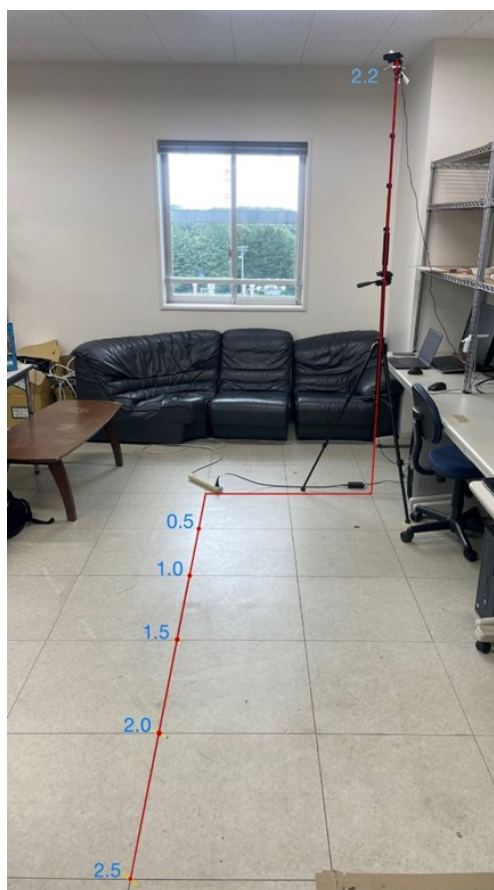


図 31 測定環境

第3章 本システムに用いる認識とポーズ判定(吉岡担当)

我々は、TSBを実現させるために、画像処理による手認識を用いる。ここで、本研究では、手認識を「手の骨格検出とポーズ判定を組み合わせたもの」と定義する。本章では、手の骨格検出及びポーズ判定の詳細について記す。

3.1 手の骨格検出

本節では、手の骨格検出について説明する。手の骨格検出とは、MediaPipe の提供するソリューションの Hands[3-1]を用い、手の骨格点座標を推定することである。MediaPipe では、手に対して WRIST(手首)や THUMB_TIP(親指先)といった 21 個のランドマークポイントが設定されており、このランドマークポイントの座標が推定される。実際に、手の骨格を表示させる MediaPipe Hands のサンプルプログラムを実行した様子を図 33 に示す。図 32 に示された骨格点座標が正しく推定されていることが分かる。

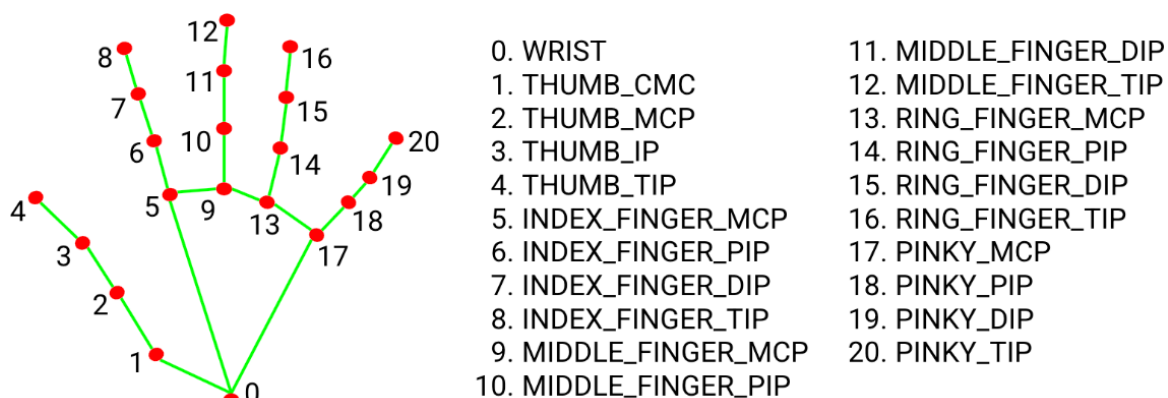


図 32 MediaPipe が定義する手のランドマークポイント [3-1]

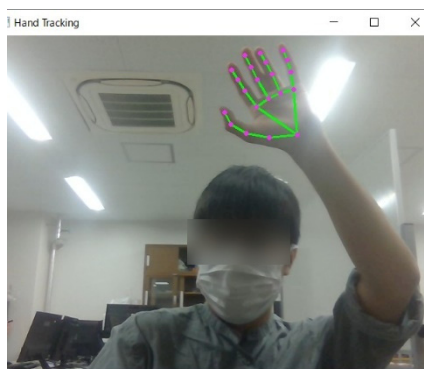


図 33 Hands のサンプルプログラムを実行した様子

3.2 ポーズ判定

本節では、手認識を「手の骨格検出+ポーズ判定」と定義した時の「ポーズ判定」について解説する。ポーズ判定とは、ニューラルネットワークを用いてユーザーが行っている手の形を判別することである。例えば、ユーザーが、図 34 のような、じゃんけんの「パー」のポーズである手を開いた動作を行った時、Open と判定されるというものである。ポーズ判定を行うためには、あらかじめ手のポーズが学習されている必要がある。これを以下では、「ポーズ学習」と呼ぶ。次項では、Kazuhito00 氏の GitHub ページ[3-2]を元にしたポーズ学習の方法を示す。



図 34 手のポーズ判定を行った様子

3.2.1 ポーズ学習

本項では、ポーズ学習の手順を説明する[3-2]。ポーズ学習には、学習元となるデータが必要である。学習のデータはExcelファイルにxyの座標と割り当てたいid、ポーズの名称(ラベル)を記述して作成する。xyデータの座標は、MediaPipeのランドマークポイントのものである。これに学習することによりお手本となるポーズと紐付けられたidとポーズのラベルが出力されるようになる。例えば、パーのポーズのidを0、ラベルをOpenとして学習を行うと、図34のようにidとラベルがそれぞれ0、Openとして認識されるのである。

3.2.2 学習の手順

本節では、学習データの追加(変更)とトレーニング手順の説明を行う。



1. キーボードの「k」キーを押し、キーポイント保存モードに変更する。
2. 画面内で学習させるポーズをとった上で、「0」～「9」のキーを押してcsvにキーポイントを保存する。なお、押したキーの数字はクラス分類に使用される。
3. Jupyter Notebookで学習を行う。
4. 学習後は新たに学習した項目が反映される。

以上の手順は、Kazuhito00氏のプログラム[3-2]を利用して行う。

3.2.3 研究初期のポーズ

本項では、研究初期に使用していたポーズについて説明する。研究初期には、表 6 のような、手を開いた状態の「Open」、手を閉じた状態の「Close」を使用していた。この 2 つのポーズを使用していた理由は 2 つある。一つ目は、Open と Close の手の形は、有名な遊戯であるじゃんけんの「パー」と「グー」と同じ形であり、ユーザーにとって馴染み深いポーズであると考えたためである。二つ目は、Open と Close はどちらも単純な手の形であるため、バス降車の際に咄嗟にポーズを行うことが出来ると考えたからである。しかしながら、この 2 つのポーズには、一点、致命的な問題点があった。次項では、その問題点を示す。

表 6 研究初期のポーズ案

	
Open	Close

3.2.4 問題点

手認識に用いているポーズは、ユーザーが咄嗟にポーズ動作を行えるよう Open や Close と言った単純なポーズを用いている。しかしながら、この二つのポーズは、日常的に行われている動作であるため、使用者の意図しないタイミングで手の認識がされ、誤認識が多発してしまう。例えば、図 35 のように手すりを掴む動作を行った際に、検出結果として Close が出力されてしまう。この問題を解決するために、Open と Close 以外の新たなポーズを選定する必要がある。この問題の改善は 4.8 で行う。



図 35 誤認識が起こり得る場面

第4章 本システム実用化のための工夫(國吉、溝口担当)

既に述べたように、本研究のプログラムは Kazuhito00 氏の作成したポーズ判定プログラム[3-2]をベースに開発されている。その際、TSBに適用するためいくつかの機能を追加する必要があった。本章ではそれらについて解説する。

4.1 複数カメラへの対応

2.5 節で述べたように、TSB の実現には 5 個のカメラをバスに設置する必要がある。Kazuhito00 氏のプログラム[3-2]はカメラ 1 つでの動作を想定している。これを 5 個のカメラで接続できるようにするよう春日井優氏の複数カメラを OpenCV で利用するプログラム[4-1]をもとに拡張する。3つのカメラを接続してプログラムを動作させている様子が図 36 である。

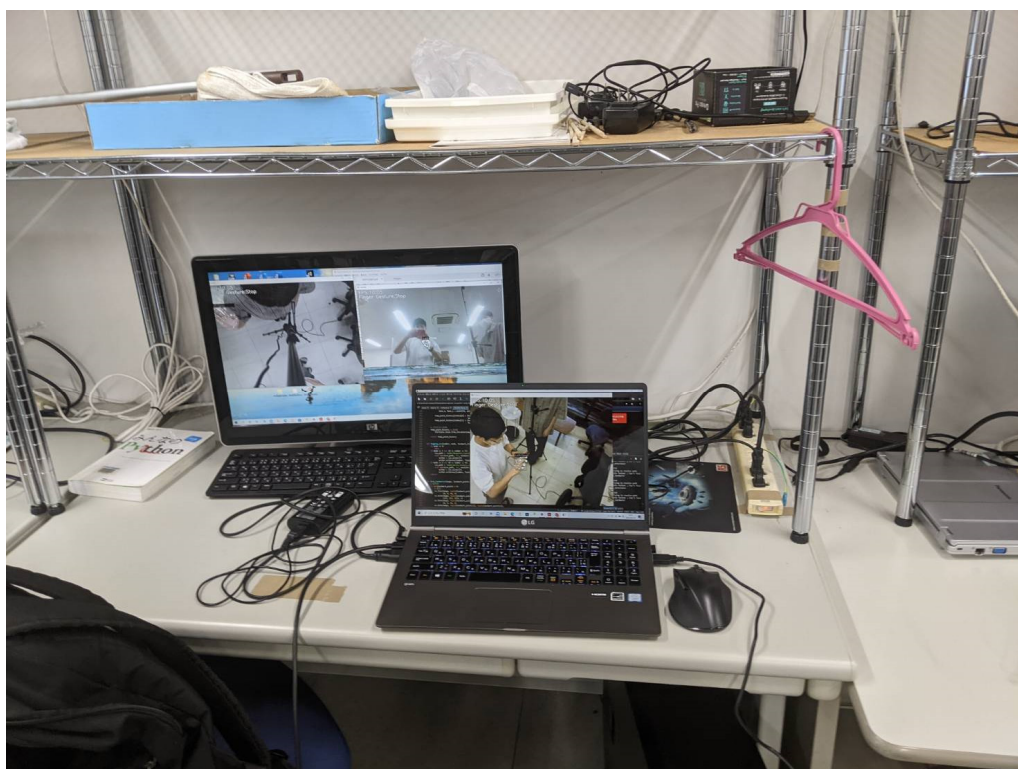


図 36 3つのカメラを用いた動作の様子

4.2 2段階化

3.2.4 で述べたように、手のポーズ認識には誤認識の問題がある。本質的な改善は 4.8 で行うが、ここではそれとは別の改善策を紹介する。

我々は図 37 のように認識を 2 段階で実現する。認識を 2 段階化することで、1 段階目に意図せず降車用ポーズをとってしまった場合でも、降車の意思があると判定されることを防ぐことが可能になる。動作手順はコード 1 の通りである。リストに 0 が格納されていた場合(1 段階目であることを意味する)で、ポーズが 0 の場合(Open であることを意味する)、段階移行の処理が行われる。そしてリストに 1 が格納されていた場合 (2 段階目) でポーズが 1 の場合 (Close)、降車の意思があると判定される、という仕組みである。

```
if 0 in list_step:
    if hand_sign_id == 0:
        print("第2段階に移行します")
        list_step[0]=1
    else:
        print("第1段階に戻ります")
if 1 in list_step:
    if hand_sign_id == 1:
        print("降車します")
    else:
        print("第1段階に戻ります")
        list_step[0]=0
```

コード 12 段階化

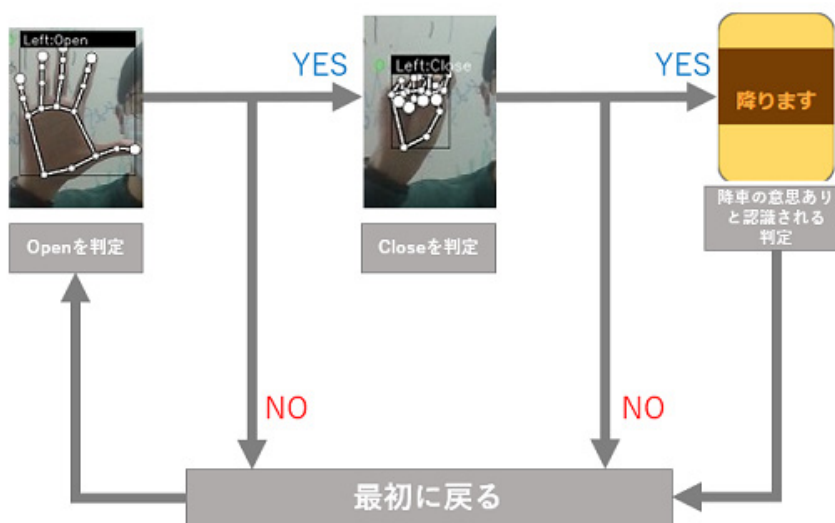


図 37 2 段階のポーズ判定による降車の判定

4.3 認識結果の処理

4.3.1 最頻値ポーズの導入

研究を進める中で、認識が一時的に安定しない現象が起こった。具体的には、手認識の判定結果が使用者の意図したものと異なる現象や認識が、途切れ途切れに行われてしまう現象である。この現象は、手認識を1フレーム当たり数10ミリ秒ごとに行い、ポーズを識別しているため起こると考えられる(図38)。その解決にはポーズの判定を毎フレーム毎に行うのではなく、複数フレームの結果をまとめてポーズを判定する必要があると考えた。そこで我々は、認識データを一定数収集し、収集したデータの中から最も多いポーズ(最頻ポーズ)を結果として用いることにした。最頻ポーズの取得は大きく分けて3つの段階で構成されている。まず初めに手のポーズを認識する。次に、その結果を200回蓄積する。そして、200回分の認識結果の中から最も多く認識されたポーズを最頻ポーズとして出力する。これにより、安定しない瞬間があった場合でも、認識結果に悪い影響を及ぼすことなく動作することが可能になった。動作手順はコード2の通りである。リストに0が格納されていた場合(1段階目であることを意味する)、1段階目のリストにポーズを格納する。2のリストにポーズが200個格納されたら、200個のポーズの中から最頻ポーズを取得する。最頻ポーズが0の場合(Openであることを意味する)、段階移行の処理が行われる。

```
if 0 in list_step:
    list_hdc.append(hand_sign_id)
    if len(list_hdc) == 50:
        cd = statistics.mode(list_hdc)
        if cd == 0:
            print("最頻値は" + str(cd) + "。第2段階に移行します")
            list_step[0]=1
        else:
            print("最頻値が" + str(cd) + "でしたので第1段階に戻ります")
            list_hdc.clear()
if 1 in list_step:
    list_hdc2.append(hand_sign_id)
    if len(list_hdc2) == 50:
        cd2 = statistics.mode(list_hdc2)
        if cd2 == 1:
            print("最頻値は" + str(cd2) + "。降車します")
        else:
            print("最頻値が" + str(cd2) + "でしたので第1段階に戻ります")
            list_step[0]=0
            list_hdc2.clear()
```

コード2 最頻ポーズ



図 38 認識が不安定な際のイメージ

4.3.2 収集データ数の削減

前項の対策の結果、安定した結果を出力することは可能になった。しかし、200 個のデータを収集するためには、カメラのフレーム数を 20FPS と仮定した場合、1 段階目の処理に 10 秒、2 段階目まで処理を行った場合は 20 秒必要になる。従来の物理的な降車ボタンであれば、1 秒未満で行える操作が 20 秒かかってしまう。そのようなシステムは非常にユーザビリティが悪く、降車ボタンの代替にはなりえない。そこで、収集するデータ数を 20 個にすることにした。それにより 2 段階処理が最短 2 秒で完了するようになった。

4.3.3 移動最頻ポーズの取得

200 個のデータから最頻値として選ばれる場合は最低でも 101 個同じポーズが必要になり、20 個の場合は最低 11 個の同じポーズが必要になる。11 個のデータを収集するのにかかる時間は、カメラのフレーム数を 20FPS と仮定した場合、最短で約 0.6 秒同じポーズを取り続ければ処理が完了する。しかし、データを 20 個収集するためのフレームが固定されていれば、より長い時間がかかることがあり得る。この時間を少しでも短くするため、我々はポーズの取得方法を移動最頻ポーズに変更した。データ処理のイメージを図 39 に表した。図上部が移動最頻ポーズ取得処理、図下部が 4.3.2 までのポーズ取得処理を表している。なお、グレーの長方形が一回の処理、長方形内の赤丸は次のフレームで破棄されるポーズ、青四角が新たに追加されたポーズを表している。図のように、1 フレーム毎に処理を行う移動最頻ポーズ取得は 4.3.2 までの最頻ポーズ取得処理と比較して処理が速いことが分かる。動作手順はコード 3 の通りである。リストに 0 が格納されていた場合(2 段階目であることを意味する)、2 段階目のリストにポーズを格納する。2 のリストにポーズが 20 個格納されたら、20 個のポーズの中から最頻ポーズを取得する。最頻ポーズ

ズが 0 の場合(Open であることを意味する)、段階移行の処理が行われる。最頻ポーズが 1 であった場合(Close であることを意味する)はリストに格納されている最初のポーズを破棄しリストに格納されているポーズの数を維持する。

以上で移動最頻ポーズの取得が実現される。

```

if list_step_0 == 0:
    list_hdc_0.append(hand_sign_id)
    if len(list_hdc_0) == 20:
        data = collections.Counter(list_hdc_0).most_common()[0][0]
        # 「パー」なら
        if data == 0:
            print(f'Cam{i}:第2段階に移行します')
            list_step_0 = 1
            list_hdc_0.clear()
        else:
            print(f'Cam{i}:認識を継続します')
            list_hdc_0.pop(0)

```

コード 3 移動最頻ポーズ

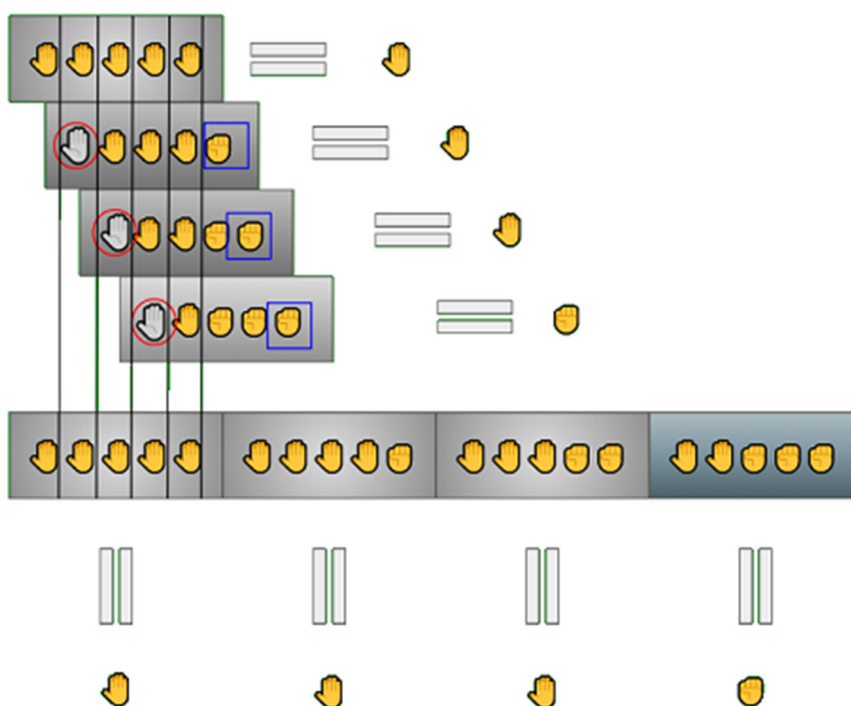


図 39 処理方法の比較

4.4 ユーザーによる処理時間の差異

4.2 の図 37 で示したように、2 段階化された認識において 1 段階目で「Open」と認識できなかった場合は最初の処理に戻り、「Open」と認識できるまで毎フレームごとに認識の判定が行われる。しかし、2 段階目で「Close」の判定に失敗した場合は、図 37 のように「Close」と認識されるまで認識の判定を行わず最初の処理に戻る。最初の処理に戻す理由は、1 段階目は無意識のうち「Open」をしていたとしても、2 段階目で降車ボタンの誤作動を起こさせないためである。動作手順はコード 4 である。リストに 1 が格納されていた場合(2 段階目に移行したことを意味する)、2 段階目のリストにポーズを格納する。リストにポーズが 20 個格納されたら、20 個のポーズの中から最頻ポーズを取得する。最頻ポーズが 1 の場合(Close であることを意味する)、段階移行の処理が行われる。最頻ポーズが 0 であった場合(Open であることを意味する)は 1 段階目に移行する。

```
if list_step_0 == 1:
    list_hdc_0.append(hand_sign_id)
    if len(list_hdc_0) == 20:
        data = collections.Counter(list_hdc_0).most_common()[0][0]
        if data == 1:
            list_step_0 = 1
            print(f'Cam[i]:降車します。')
        else:
            print(f'Cam[i]:第1段階に戻ります')
            list_hdc_0.clear()
            list_step_0 = 0
```

コード 4 変更前

しかし、この方法では 2 段階目に移行した直後、リストには 1 段階目のポーズが多く含まれるため認識に失敗する時間が存在する。この時間は 2 段階目のポーズに切り替えるのにかかる時間(反応速度)に依存するため個人差がある(図 40)。この問題を、2 段階目のポーズ収集数を 1 段階目の 2 倍の 40 個、あるいは 3 倍の 60 個に変更して動作時間に余裕を持たせることで解決することはできる。しかし、使用者は反応速度が速い人から遅い人まで多種多様であると予測されるため、この手法では反応が速い人は遅い人に合わせることになる。

そのため、2 段階目の処理の待ち時間を可変化した。そのイメージ図が図 40 である。可変処理のためには、1 段階目終了後から経過した時刻を計測する必要がある。5 秒以内に 2 段階目のポーズをとれば成功と見なす。5 秒以内は「Close」と認識できるまで毎フレームごとに認識の判定が行われ、判定完了の時刻は人によって異なる。そのため、使用者の反応速度に合わせた処理が可能になったと言える。動作手順はコード 5 の通りである。5 秒経過しない限り、ポーズの

格納、最頻ポーズの取得、最頻ポーズが2段階のポーズと一致しているかの確認が継続して行われる。時間差が5秒になると1段階目に移行する。

```

if step_0_0 == 'Open':
    diff_time = switch_time_0_0 - remaining_time
    if not diff_time == 5:
        list_hdc_0_0.append(hand_sign_id)
        if len(list_hdc_0_0) == 20:
            data = collections.Counter(list_hdc_0_0).most_common()[0][0]
            if data == 4:
                check = 1
                s = sound2()
                s.start()
            else:
                list_hdc_0_0.pop(0)
    else:
        list_hdc_0_0.clear()
        step_0_0 = 'Close'
        switch_time_0_0 = 0
    
```

コード5 変更後

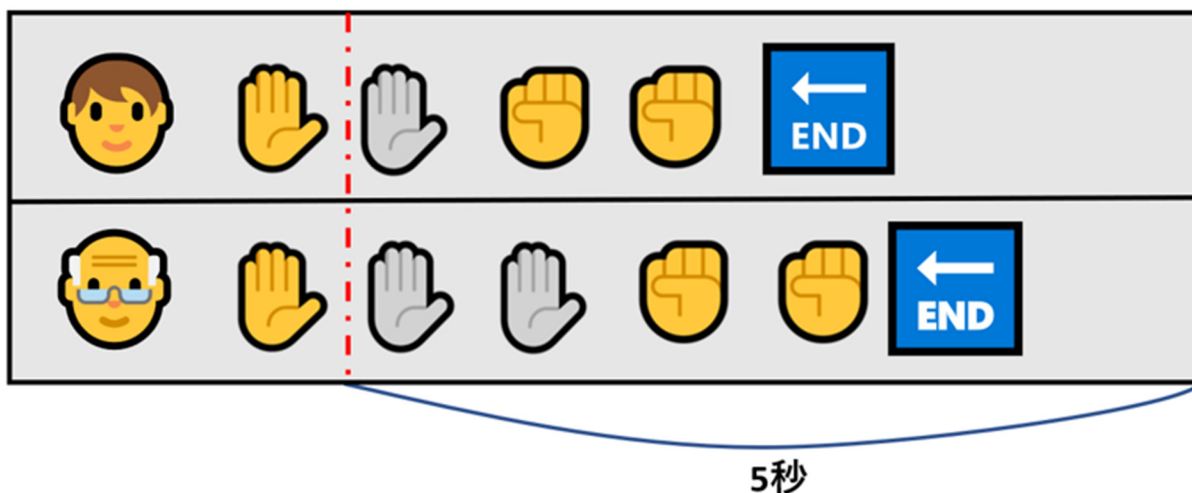


図 40 可変処理のイメージ図

4.5 複数個の手の認識

次に、複数人認識(複数個の手の認識)について解説する。4.4 節までの TSB は、1 個の手までの動作にのみ対応しており、複数個の手を認識させると問題が発生した。その問題とは複数個の手が 1 個の手として処理されてしまうことである。例えば、図 41 のように 2 人の使用者がポーズをしている場合を考える。ポーズ判定用に用意されるリストは 1 人分までしか用意されておらず、図 42 のように 1 つ分のリストの中に 2 人分のデータが追加されていた。そのため、意図した認識結果が得られない。この問題を解決するために、まず手に番号をつけて分類した。そして、その番号に対応したリストを用意した。そのイメージを図 43 に示す。これにより、複数人の手を別々に認識できるようになる。

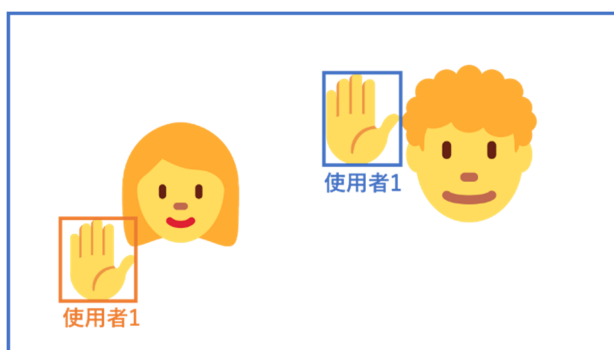


図 41 複数の手の認識



図 42 複数人の手のポーズが蓄積される問題

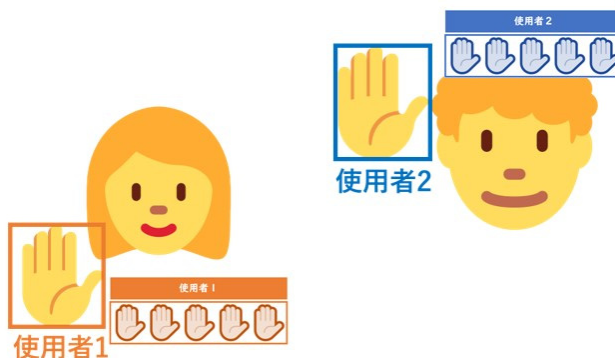


図 43 複数個の手の認識のイメージ

```
list_old = []
```

```
hand_id = 0
```

```
if hand_id == 0:
```

```
if len(list_old) != 0:
```

```
if len(results.multi_hand_landmarks) != list_old[-1]:
```

```
    print("hand_id の変化を検知したため、初期化します")
```

```
list_old.pop(0)
```

```
list_old.append(len(results.multi_hand_landmarks))
```

```
hand_id += 1
```

赤線は記述省略を意味する

コード 6 手の分類

最初から手に整数値が定義されているわけではないので、整数型(hand_id)(コード 6)として定義し、for 文の繰り返し(コード 7)を利用する。

```
for hand_landmarks, handedness in zip(results.multi_hand_landmarks,  
                                     results.multi_handedness):
```

コード 7 繰り返し処理の部分

繰り返した回数分 hand_id が増える処理を追加した。これにより、手の個数分の hand_id が割り振られるようになる。しかし、この ID は影響範囲内の手の個数が変化したときに値が変わってしまうという問題があった。そこで、手の個数が変化した場合を results.multi_hand_landmarks の値の要素数の変化から判断し、認識をやり直す処理を追加した。これにより、手が頻繁に変化するような場面でも認識を安定して行うことができる。





4.6 複数人の手の骨格検出

前節では、複数人の手のポーズ認識について説明を行った。本節では複数人の手の骨格の検出について説明を行う。

4.6.1 複数人の手の検出の検証

ここで、当初の目標であった 10 人(10 個)の手での動作が行えるか検証を行った。主な検証方法は、Kazuhiro00 氏のプログラム[3-2]に含まれる min-tracking-confidence(mdc)という手の骨格検出の甘さを設定する項目を 0.5 から 0.9 まで変更するというものである。mdc は数値が小さいほど手の検出の精度が甘くなり、手の検出が容易になる。しかしながら、精度が甘くなってしまったために、人間の手とは全く別の物が手と認識されてしまうことが起きてしまった。本検証では、全く同じ手を 12 個並べた画像を使用する。その画像をカメラに映し、誤認識の少ない mdc の値を求める。検証を行った結果を表 7 に示す。

表 7 mdc と手の個数の関係

	
mdc=0.9 手の個数 9/12	mdc=0.8 手の個数 12/12
	
mdc=0.7 手の個数 11/12	mdc=0.6 手の個数 12/12



mdc=0.5 手の個数 12/12

検証の結果 mdc の値が 0.8、0.6、0.5 の時、12 個全ての手を検出することが出来た。しかしながら、mdc=0.8 が 12 個全て読み取れたことに対し、mdc=0.8 よりも精度が甘いはずの mdc=0.7 では、12 個の手を検出することが出来なかった。そのため、今後は安定性を求めて、mdc=0.8 ではなく mdc=0.6 の値を使用することに決定した。

4.6.2 Frames Per Second 計測

本節では、1 秒間に描画されるフレームの単位である Frames Per Second(FPS)の計測を行い、手の個数が与えるパフォーマンスへの影響を調査した。調査の方法は、手が複数 1~10 個まで写っている状況を再現し、それぞれの個数における FPS を計測した。計測はポーズ判定が始まった瞬間から計測されている。1 段階目の認識が終了するまでの平均値をそれぞれの手の個数における FPS の値としている。FPS の値の推移をグラフで表す。

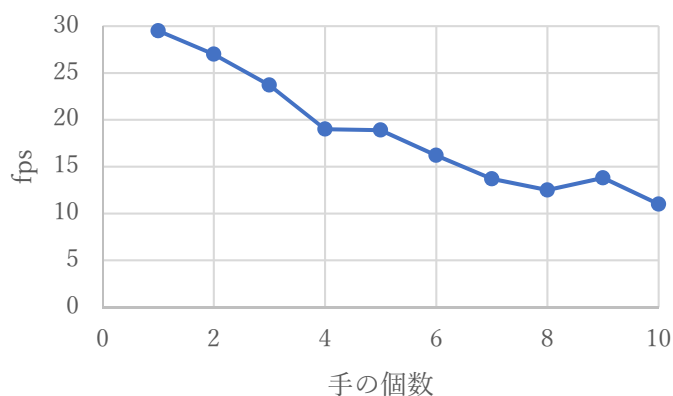


図 44 手の個数と FPS の推移

図 44 より、手の個数が増えると FPS が低下することが分かる。このデータの相関係数は-0.931 となり、強い負の相関がある。

4.7 誤認識の低減

4.7.1 問題点

システムを動作させた場合、ポーズの誤認識が発生する。その問題について解説する。例えば Open と Close のポーズのみで認識を行うと、あらゆる手のポーズは Open か Close かのどちらかと認識されてしまう。すなわち、手が Open か Close 以外の状況に対応できない。

4.7.2 誤認識防止ポーズの追加

この問題の解決のため、複数の誤認識防止ポーズを追加した。Open と Close 以外の様々な手のポーズをいくつか想定し、それらを一つ一つ別々のクラスとして学習させるのである。そして、Open と Close 以外のポーズが認識された場合にそれらを「誤認識防止ポーズ」に分類することにする。

4.7.3 検証及び結果

検証には、動作に必要なポーズである Open(図 45)と Close(図 46)及び誤認識防止ポーズとして phone(図 47)と grip(図 48)を用意した。つり革やスマートフォンや傘等、握る動作は誤認識につながる可能性が高い。検証では追加前と追加後のシステム両方で4つのポーズを行った。それぞれのポーズが間違いなく判定できていた場合を成功とし、成功率を比較した。



図 45 Open



図 46 Close



図 47 phone



図 48 grip

表 8 成功率

	追加前	追加後
成功率	50%	97.5%

表 8 のとおり学習後では曖昧な手を適切にその他のポーズとして分類できており、誤認識が減少した。

4.8 ポーズの改善

本節では、3.2.4で解説した誤認識の問題について述べる。この問題は4.7で解説した誤認識防止ポーズの追加でもある程度改善できるが、ポーズの変更によりさらなる改善を狙う。

4.8.1 Natural User Interface

3.2.4の問題を解決するために、Natural User Interface(NUI)[4-2]を用いてポーズを変更する。NUIはマイクロソフトが提唱するジェスチャやポーズのガイドラインである。NUIには様々な項目があるが、その中でもTSBのポーズとして用いることができる要素を表9にまとめた。

表9 採用したNUIの評価項目

1	意味が適切か
2	直感的か
3	距離に影響しない
4	シンプルである
5	学習しやすいか
6	類似してないか
7	片手でできるか
8	学習できるか

4.8.2 アンケート及び結果

前項のNUIを参考に選定したポーズは図49の4種となった。従来までのOpenとCloseに加えて、GoodとPeaceが新たな候補となった。この中からポーズを選ぶアンケートを20代前半の男性8名に対して行った。アンケートは1人2票である。アンケート結果を図50に示す。

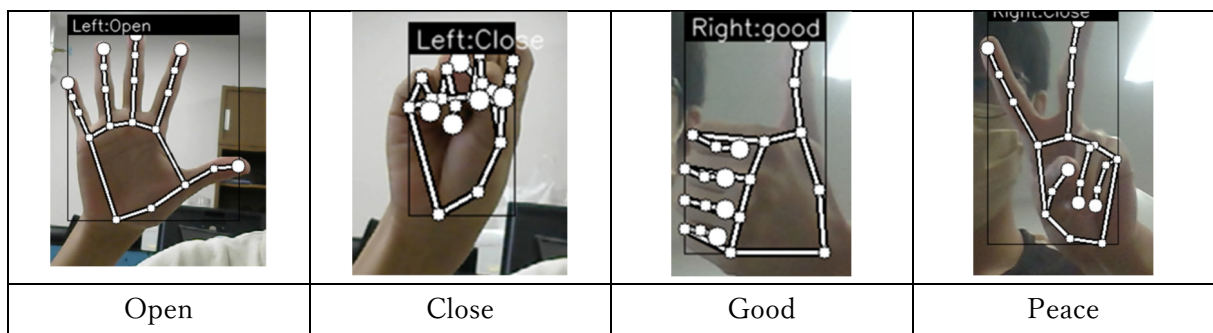


図49 選定候補のポーズ

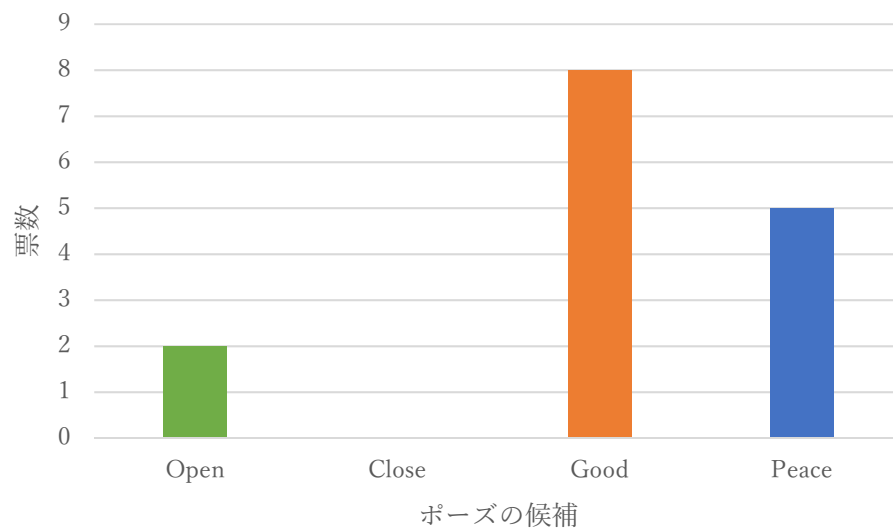


図 50 アンケート結果

アンケートの結果から、上位 2 位となった Good と Peace のポーズを採用する。なお、ポーズを 2 つ選ぶ理由は、本研究では手認識に 2 段階認識を用いるからである。

4.9 フィードバック

TSB は降車ボタンと異なり、使用者への直感的なフィードバックがないため、操作に対して処理が行われたかを判断する手段がない。特に 2 段階目への移行時のフィードバックがないため、使用者が 2 段階目のポーズに変更するタイミングをつかめず、TSB が正常に動作しないことがある。そこで、TSB にフィードバックを用意する必要がある。フィードバックの例としては航空機の失速警報装置[4-3]がある。これは、音、光、操縦桿の振動で失速をパイロットに伝えている。このように、5 感の中で聴覚、視覚、触覚がフィードバックとして用いられているものが多い。我々が普段利用する路線バスで降車ボタンを押すと、案内放送の再生、車内表示器の表示、降車ボタンに付属する電球の点灯、以上 3 つのフィードバックが使用者に対して行われる。

4.9.1 バス環境の再現

実際のバスを再現するため、本学学生が多く利用する西東京バス工 01 系統のバスをシチュエーションとして使用する。シチュエーションとして使用する停留所は表 10 の通りである。本停留所番号は我々が研究の都合上定義したものであり、西東京バスが定めた番号とは異なる。なお、表番号の 13 と 14 の間には中野市民センターがあるが、本停留所は復路のみ停車するため含めないものとする。実際のバスでは次停留所の案内には様々なデータが用いられている(図 51)。我々はそれらを用意することができないため、停留所間の時間を計測することで再現した。表 11 と図 52 は、実際に往路復路各 2 回乗車し、各停留所間の時間を計測した結果をまとめたものである。

表 10 停留所

1	横山町
2	横山町三丁目
3	八日町一丁目
4	八日町四丁目
5	織物組合
6	本郷横丁
7	平岡町
8	中野上町郵便局
9	中野
10	西中野二丁目
11	西中野三丁目
12	東檜原
13	新清水橋
14	工学院大学西

表 11 停留所間の時間

	10:06 発(往路)	10:26 発(往路)	16:10 発(復路)	16:25 発(復路)
1	124.62	94.33	147.22	357.32
2	42.72	101.73	33.73	40.45
3	31.54	119.14	124.16	101.59
4	74.02	93.61	30.54	111.60
5	38.07	51.38	162.77	95.23
6	41.78	80.14	35.45	30.74
7	62.57	80.71	227.00	166.96
8	94.80	40.73	243.89	280.19
9	21.61	31.37	25.99	24.97
10	41.56	33.37	25.37	33.19
11	163.76	129.14	130.30	73.78
12	47.42	123.12	51.61	52.38
13	52.92	44.31	111.49	104.99
14	131.16	68.95	171.8	93.50



図 51 案内に用いられるデータ [4-4]

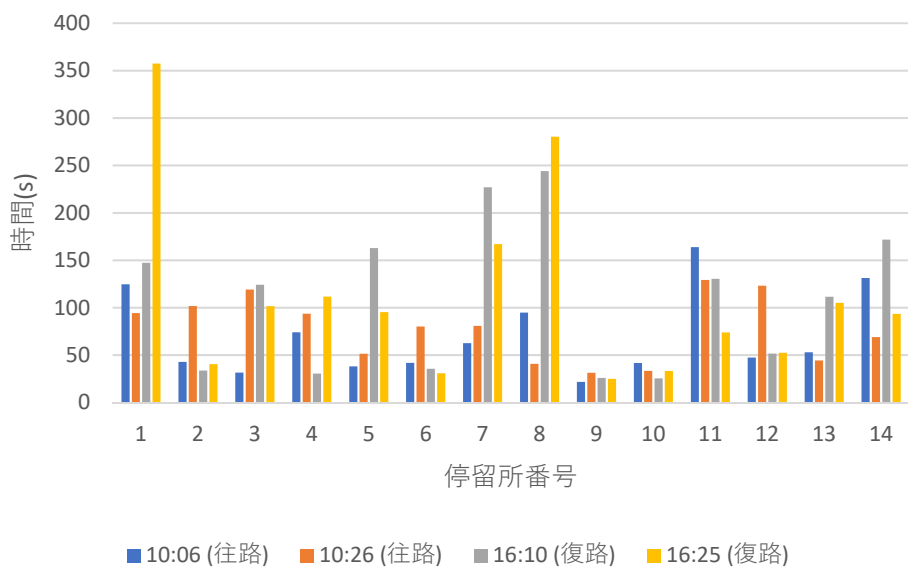


図 52 停留所間の時間

4.9.2 実装したフィードバック

我々は使用者へのフィードバックとして、5 感の中から聴覚と視覚を用いることにする。下記に実装した聴覚と視覚のフィードバックを示す。

聴覚フィードバック

聴覚のフィードバックシステムとして、段階が変更した場合と降車が決定した際に音を鳴らすフィードバックを実装した。デバッグ中、音声再生時に画面が乱れる現象が起きたが、この現象は音声再生中に他処理が一時停止してしまうことから発生していると判明した。そこで、音声を再生する部分と画面表示の部分のスレッドを分け、同じ CPU コアの同じ python プロセスで複数の処理を同時に行う、音声再生部を並行処理に変更したところ問題が解決した。

我々のシステムでは、2 段階目に移行した際と 2 段階目の認識が成功した際、次停留所案内で聴覚フィードバックを用いる。各フィードバックのサウンドは表 12 の通りである。なお、音声はヒホ氏が開発したテキスト読み上げ用音声合成ソフトウェア、「VOICEVOX」に収録されているキャラクター、「四国めたん」の声を使用し作成した(図 53)[4-5]。

表 12 フィードバックサウンド一覧

2 段階目に移行した際	「テレン」という効果音
2 段階目の認識が成功した際	「次、とまります」という音声
次停留所案内	「次は、〇〇、〇〇です」という音声

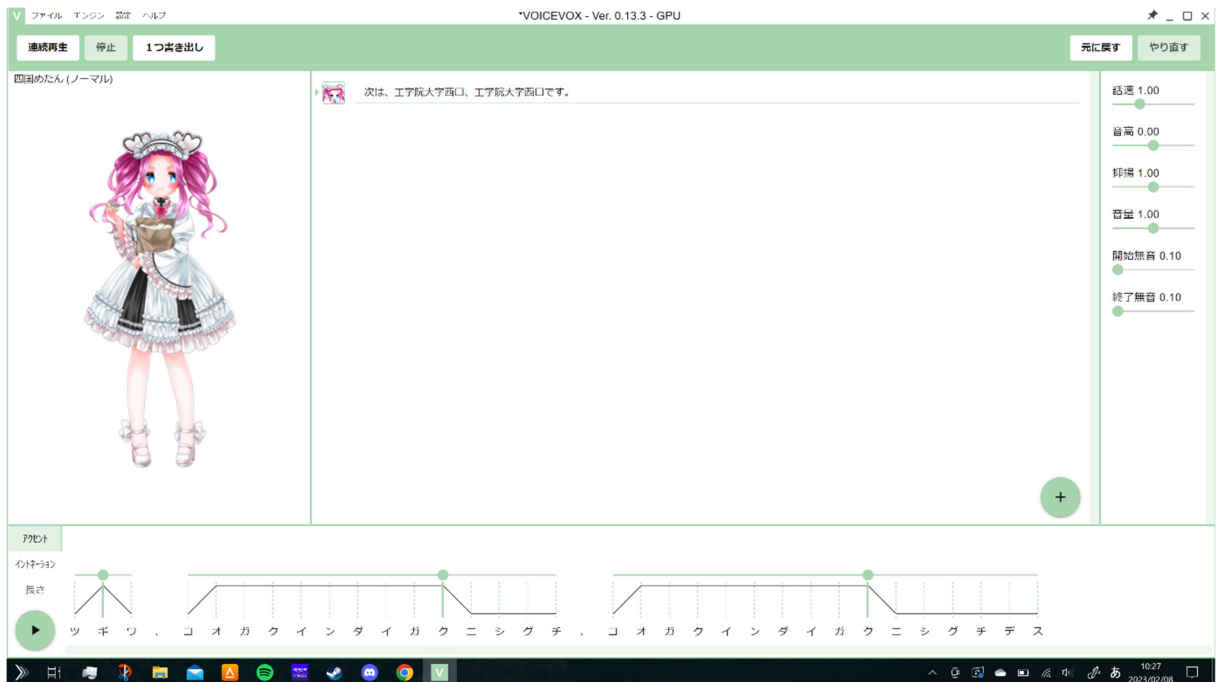


図 53 VOICEVOX

視覚フィードバック

視覚のフィードバックとして、文字を開発者画面内に追加した。なお文字は OpenCV の仕様上、日本語に対応していないため英語で表記している。画像上部に次の停留所、その隣に停留所通過までの残り時間が表示される。ただし、視覚フィードバックをバス車内に映すことは、プライバシーの観点から望ましくない。そのため、現時点では視覚フィードバックの内容は開発者専用としている。

4.9.3 開発者画面の動作内容

前項 2 つのフィードバックが適用された場合の動作を図 54～図 57 を用いて解説する。何もポーズを行っていない場合は、図 54 のようにフレームレートと 4.9.1 項で述べた「次の停留所名及び停留所までの残り時間」が表示される。残り時間が 0 秒になったタイミングで停留所名と残り時間は更新される。また同タイミングで 4.9.2 項記載の、聴覚フィードバックが再生される。

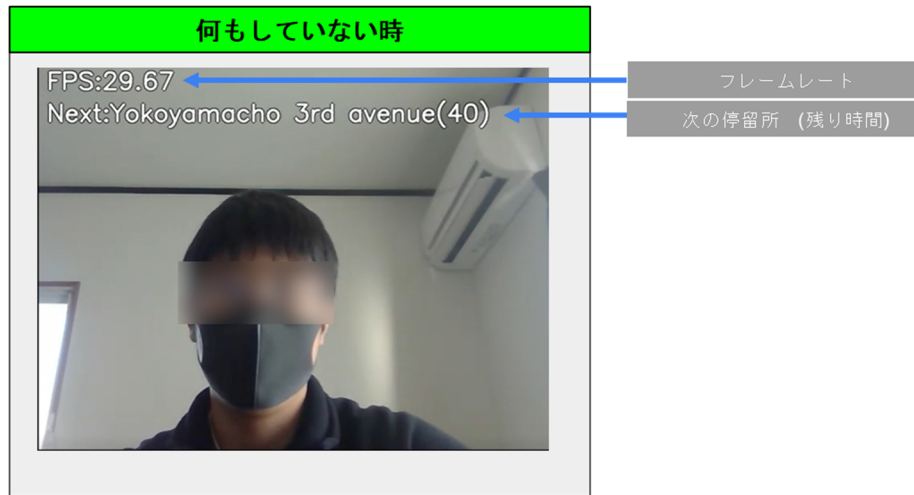


図 54 動作開始時の画面

ポーズを行った場合は図 55 のように、先ほどの情報に加えて手に対しての情報が描写される。表示される内容は3つある。1つ目は手を識別する ID である。2つ目は、1段階目クリアに必要なポーズ(左)と現在のポーズ(右)である。3つ目はポーズの状態を示す枠であり、一致すると緑色に描写される。



図 55 1段階目のポーズ認識中の様子

図 55 の状態を継続すると図 56 のようになり、合わせて 4.9.2 項記載した 2段階目への移行時の効果音が鳴る。その後、1段階目クリアに必要なポーズが 2段階目クリアに必要なポーズに更

新される。図 56 は 2 段階目に移行した瞬間のため、まだ 1 段階目クリアに必要なポーズを行っている。そのため、2 段階目クリアに必要なポーズと一致せず枠の色が赤色になる。

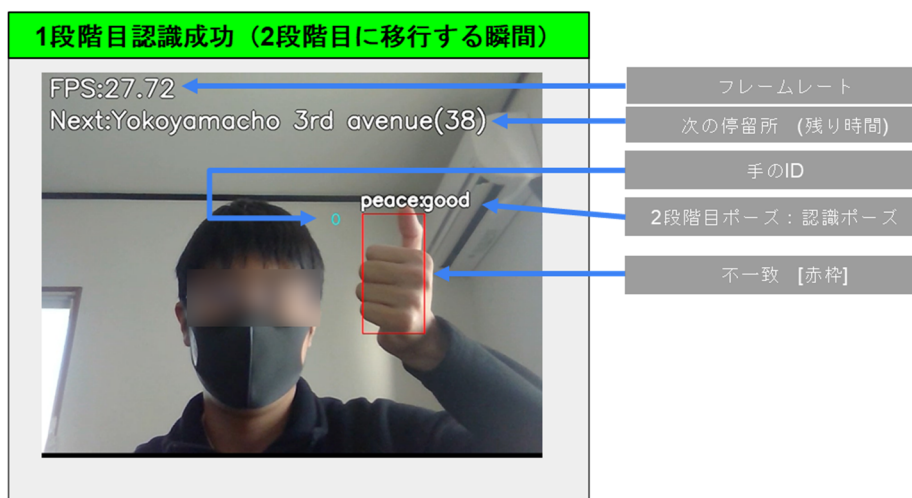


図 56 1 段階目の認識の成功の様子

図 57 のように使用者が 2 段階目のポーズをすると、2 段階目クリアに必要なポーズと一致し、枠の色が緑色となる。

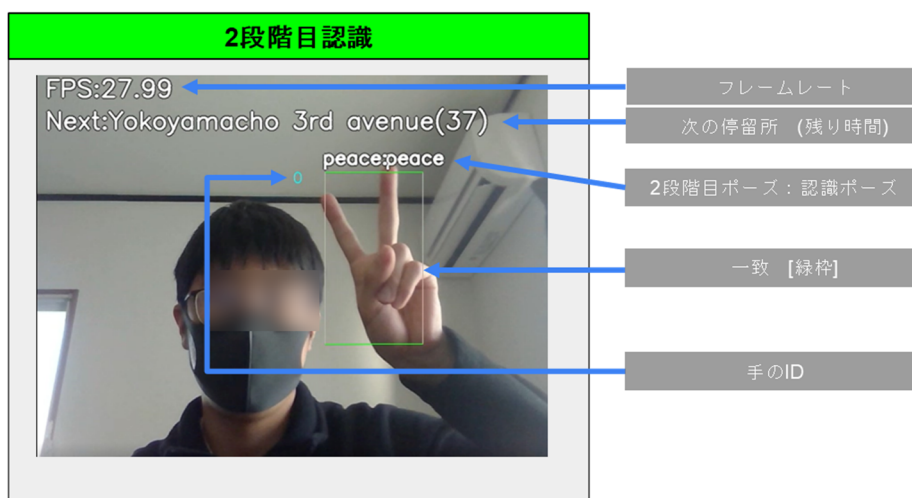


図 57 2 段階目のポーズの認識中の様子

図 57 の状態を続けると図 58 のように赤色の文字で降車合図が表示される。この状態になると音声で「次止まります」とアナウンスがされる。降車合図は次の停留所まで継続して表示され、次の停留所から再度認識を行うことができる。

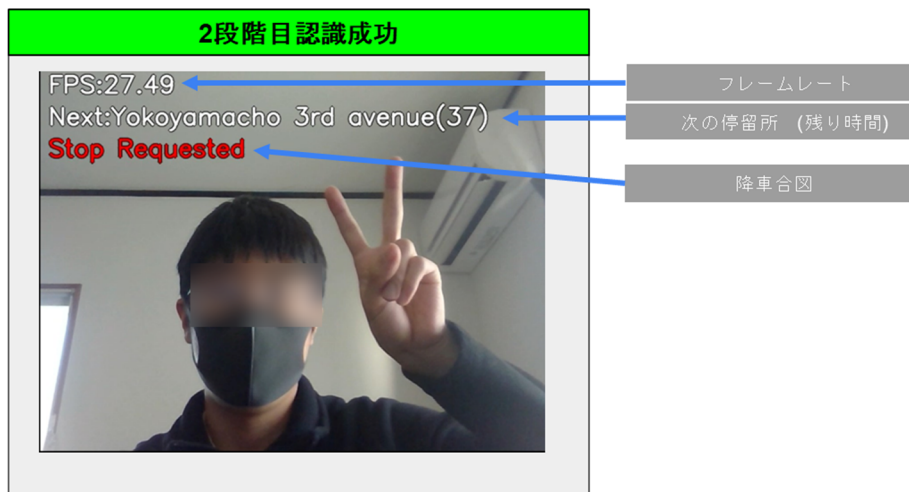


図 58 降車合図が受理された様子

段階クリアの条件を満たさないポーズを行った場合は、図 59 のようになる。ポーズの状態を表す部分のポーズが一致していないことが分かる。そのため、枠の色が赤色になり段階変更及び降車合図が行われない。

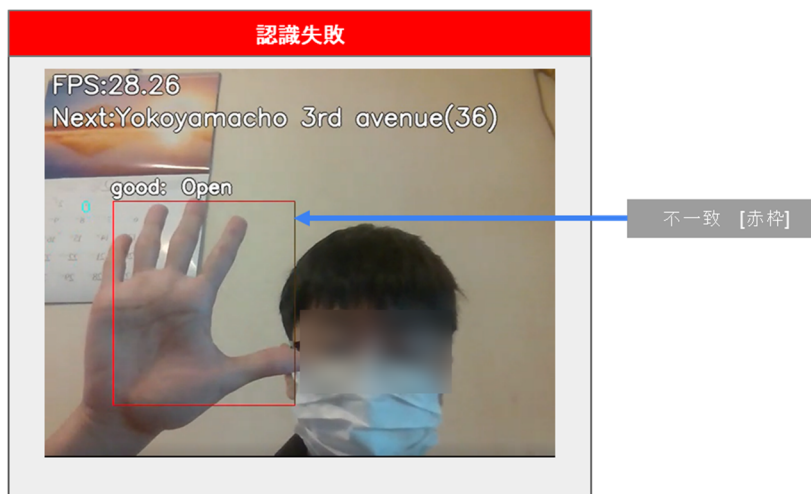


図 59 降車合図に失敗した様子

以上のことから、開発者画面での視覚フィードバックは導入に成功したと言える。

第 5 章 TSB の評価(國吉担当)

本章では実際の路線バスの環境を再現し、TSB の評価を行った。

5.1 評価方法

評価は、使用者が降車合図を 5 秒以内に出せるかどうかで行う。22~23 歳の男性 10 人の被験者に協力をしてもらい、降車合図が出るまでを 1 回とし、1 人 10 回ずつ行った。使用者は、前期に測定したバス車内の寸法とカメラ位置を基にカメラから認識できる範囲の中心でポーズを行う(図 60)。高さが地面から 2.2m で、XZ 軸角度 60° 、XY 軸角度が 45° となっている。実験の成功条件は先ほど説明した 1 段階目のポーズを Good、2 段階目のポーズである Peace が 5 秒以内に行えた場合である。

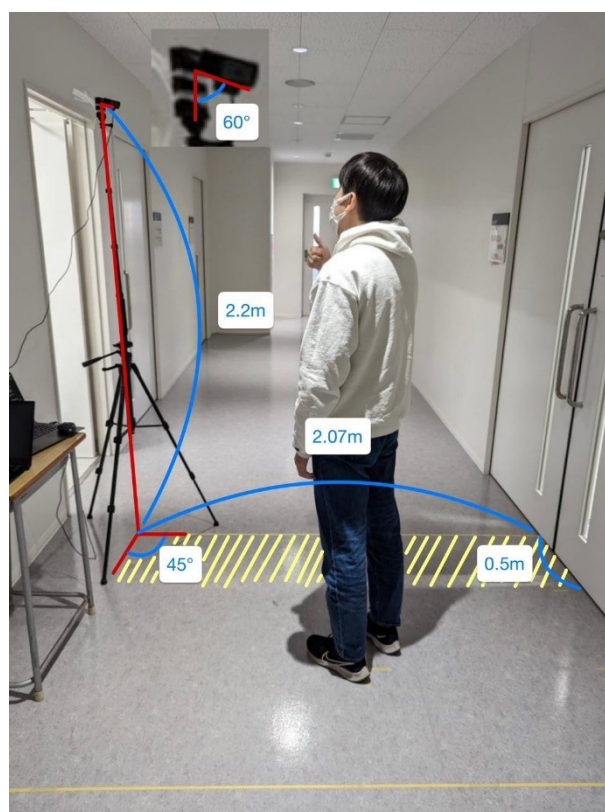


図 60 実験のイメージ図

5.2 結果

図 61 は被験者別の動作成功率である。全体の動作成功率は 98%である。我々の目標であったバス降車の意思表示の成功率 95%以上を満たす結果となった。このことから十分な成功率に達した。動作に失敗した 2%は、被験者の曖昧なポーズがクリアに必要なポーズとして認識されなかったためであると考えられる。

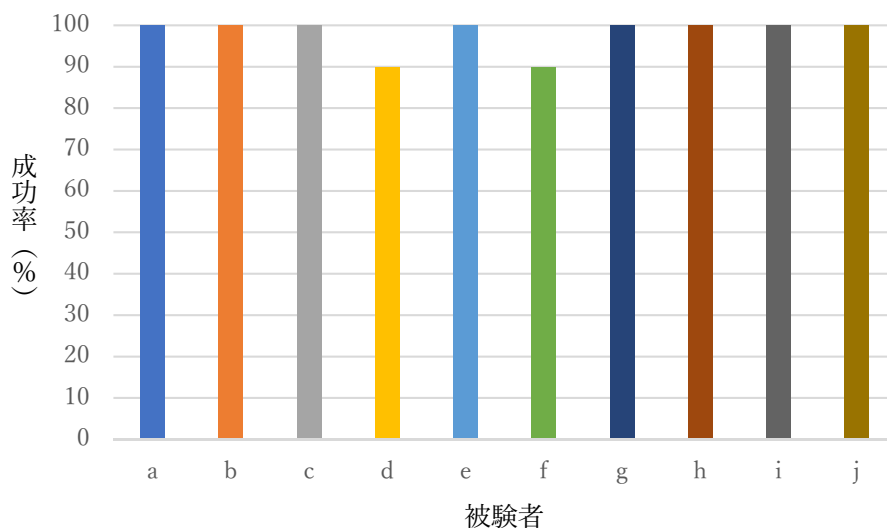


図 61 被験者別の動作成功率

5.3 課題

本実験では、22~23 歳の男性 10 人の被験者の協力により実験を行った。実験により、TSB の動作成功率が高いという結果が示された。しかし、被験者が行ったポーズが曖昧であったことに起因し誤認識する場面があった。解決には、誤認識の低減について更なる改善が必要である。また、路線バスを利用する人々は性別年齢が様々であり、手認識での反応速度がとても速い人もいれば、遅い人もいることが考えられる。そのため今後は、被験者の性別年齢ごとに、TSB の検出率を確かめる必要がある。

第 6 章 結論(國吉担当)

本研究では、バスの環境を再現した状況において、95%以上の検出率と 10 個以上の手を認識可能な「非接触式路線バス降車ボタン」の開発を目指した。まず、95%以上の検出率については、TSB の評価実験を行い、検出率は 98%であったことから条件を満たした。次に、10 個以上の手を認識する点については、手に固有の ID を割り振ることで複数個の手の認識を可能にし、mdc の値を調整することで、10 個以上の手検出が行えると確認できたため、条件を満たした。以上から、我々の掲げる到達目標を達成する非接触式路線バス降車ボタンの開発に成功した。

参考文献・URL

[1-1] NID 国立感染研究所「IDWR 2021 年第 39 号〈注目すべき感染症〉直近の新型コロナウイルス感染症の状況」<https://www.niid.go.jp/niid/ja/2019-ncov/2487-idsc/idwr-topic/10704-idwrc-2139.html> (閲覧日：2023 年 2 月 6 日)

[1-2] 厚生労働省「データからわかる-新型コロナウイルス感染者情報-」<https://covid19.mhlw.go.jp/> (閲覧日：2023 年 1 月 31 日)

[1-3] 国立感染症研究所「新型コロナウイルス(SARS-CoV-2)の感染経路について」<https://www.niid.go.jp/niid/ja/2019-ncov/2484-idsc/11053-covid19-78.html> (閲覧日：2023 年 2 月 1 日)

[1-4] 株式会社アスカネット「「コロナショック前後のモノとの接触」に関する意識調査」<https://www.asukanet.co.jp/contents/news/2020/20200525.html> (閲覧日：2023 年 2 月 1 日)

[1-5] 日本航空株式会社「本邦航空会社初、チェックイン時のタッチパネル非接触化を実施」<https://press.jal.co.jp/ja/release/202103/005996.html> (閲覧日：2023 年 2 月 1 日)

[1-6] Yan, S., Ji, Y. and Umeda (2019), K., “A system for home appliance operation by hand waving in a user-definable command space,”
<https://www.mech.chuo-u.ac.jp/umedalab/publications/paper.html> (閲覧日：2023 年 2 月 1 日)

[1-7] 顔ら (2020)「任意の位置での簡単なジェスチャによる家電操作システム」
https://www.jstage.jst.go.jp/article/transjsme/87/898/87_20-00310/_article/-char/ja/ (閲覧日：2023 年 2 月 1 日)

[1-8] 中洲ら (2013)「自然な手振りによる直感的なハンドジェスチャ UI」
https://www.jstage.jst.go.jp/article/his/15/1/15_25/_article/-char/ja/ (閲覧日：2023 年 2 月 1 日)

[1-9] 東京都「デジタルサイネージを活用した観光案内標識のサービス開始について」
<https://www.metro.tokyo.lg.jp/tosei/hodohappyo/press/2017/04/10/08.html> (閲覧日：2023年2月1日)

[1-10] レシップ株式会社「降車信号装置(押しボタン)」
<https://www.lecip.co.jp/lecip/products/bus/other01.html> (閲覧日：2023年2月1日)

[2-1] 北陸鉄道株式会社「車両紹介」<http://www.hokutetsu.co.jp/charter-bus/car-model> (閲覧日：2023年2月1日)

[2-2] 日産自動車株式会社「CIVILIAN」
<https://history.nissan.co.jp/CIVILIAN/W41/0806/function.html> (閲覧日：2023年2月1日)

[2-3] トヨタ自動車株式会社「ハイエースバン」<https://toyota.jp/hiacevan/grade/> (閲覧日：2023年2月1日)

[2-4] 西東京バス株式会社「路線型貸切バス(短時間)のご案内」
https://www.nisitokyobus.co.jp/charter/charter_rosen.html (閲覧日：2023年2月1日)

[2-5] 日野自動車株式会社「HINO Blue Ribbon / HINO Rainbow」
https://www.hino.co.jp/blueribbon_rainbow/safety-maintenance/ (閲覧日：2023年2月1日)

[2-6] 株式会社ロジクール「C920n HD PRO ウェブカメラ」<https://www.logicool.co.jp/ja-jp/products/webcams/hd-pro-webcam-c920n.960-001261.html> (閲覧日：2023年2月1日)

[2-7] 株式会社日本 HP「HP Spectre x360-15-df0009tx」<https://support.hp.com/jp-ja/product/hp-spectre-15-df0000-x360-convertible-pc/23238160/model/24575126/product-info>
(閲覧日：2023年2月1日)

[2-8] Google LLC「MediaPipe」<https://google.github.io/mediapipe/> (閲覧日：2023年2月1日)

[3-1] Google LLC 「MediaPipe Hands」

<https://google.github.io/mediapipe/solutions/hands.html> (閲覧日：2023年1月30日)

[3-2] KazuhitoTakahashi 「hand-gesture-recognition-using-mediapipe」

<https://github.com/Kazuhito00/hand-gesture-recognition-using-MediaPipe> (閲覧日：2023年2月1日)

[4-1] 春日井 優 「web カメラをたくさんつないで OpenCV で一斉に映したら監視カメラっぽくなった」 https://joho-ka.mints.ne.jp/multi-camras-with-opencv?doing_wp_cron=1675821201.9948101043701171875000

(閲覧日：2023年2月1日)

[4-2] Microsoft Corporation 「Human Interface Guidelines v2.0」

<https://download.microsoft.com/download/6/7/6/676611b4-1982-47a4-a42e-4cf84e1095a8/kinecthig.2.0.pdf> (閲覧日：2023年1月30日)

[4-3] The Boeing Company 「ANGLE OF ATTACK」

https://www.boeing.com/commercial/aeromagazine/aero_12/attack_story.html (閲覧日：2023年2月1日)

[4-4] レシップ株式会社 「運行支援システム」

<https://www.lecip.co.jp/lecip/products/bus/tms01.html> (閲覧日：2023年2月1日)

[4-5] VOICEVOX

<https://voicevox.hiroshiba.jp/> (閲覧日：2023年2月1日)

謝辞

最後に、この場をお借りして本研究を進めるにあたり、2年間多大なご指導を頂きました金丸隆志教授、そして実験に協力して頂いた皆様に心より感謝いたします。